Causal Message Sequence charts

Thomas Gazagnaire, Shaofa Yang, Loïc Hélouët, Blaise Genest, P.S. Thiagarajan

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Causal Message Sequence Charts

Thomas Gazagnaire, Shaofa Yang, Loïc Hélouët, Blaise Genest, P.S. Thiagarajan

N˚6301

Septembre 2007

Systèmes communicants

ISSN 0249-6399

Causal Message Sequence Charts

Thomas Gazagnaire, Shaofa Yang, Loïc Hélouët, Blaise Genest, P.S. Thiagarajan

Systèmes communicants
Projet DISTRIBCOM - National University of Singapore

Rapport de recherche n° 6301 — Septembre 2007 — 36 pages

Abstract: Properties of scenario languages (Message Sequence Charts, Live Sequence Charts, UML's sequence diagrams) have been widely studied in the last decade. Scenario languages embed the expressive power of Mazurkiewicz traces, and consequently, several problems such as model checking are undecidable for these languages. Despite their expressive power, most of scenario languages can only model finitely generated behaviors that can be described as the concatenation of patterns from a finite set. However, non-finitely generated behaviors such as sliding windows frequently occur in asynchronous distributed systems. Several extensions of Message Sequence Charts have been proposed to capture non-finitely generated behaviors, but these variants embed the expressive power of automata communicating via unbounded channels (and thus of Turing Machines), making their analysis even more intractable. We propose a new extension of Message Sequence Charts that can model non-finitely generated MSC languages without embedding the expressive power of communicating automata, and study its properties.

Key-words: Scenario languages, partial orders, distributed systems, modeling

(Résumé : tsvp)

This work was supported by the CASDS INRIA associated team

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Message Sequence Charts Causaux

Résumé : Les propriétés des langages de scénarios (Message Sequence Charts, Live Sequence Charts, ou diagrammes de séquences d'UML) ont été largement étudiées ces dix dernières années. Les langages de scénarios ayant la puissance d'expression des traces de Mazurkiewicz, de nombreux problèmes pratiques, comme le model-checking, sont indécidables pour ces langages. De plus, malgré leur puissance d'expression élevée, la plupart des langages de scénarios ne permet de générer que des comportements finiment engendrés, qui ne peuvent être exprimés que comme des concaténations séquentielles de comportements choisis dans un ensemble fini de motifs. Cependant, on rencontre couramment des comportements non-finiment engendrés lorsque l'on cherche à modéliser des systèmes distribués et asynchrones, ce qui est le cas du mécanisme de fenêtres glissantes du protocole TCP. Plusieurs extensions des Message Sequence Charts ont été proposées pour remédier à ce problème, mais ces variantes donnent aux scénarios la puissance d'expression des automates communicants à canaux de communication non-bornés (et par conséquent des Machines de Turing), rendant l'analyse de ces langages encore plus difficile. Ce rapport propose une extension des Message Sequence Charts qui permet de définir des comportements non finiment engendrés sans pour autant donner aux scénarios la puissance des automates communicants, et étudie ses propriétés.

Mots-clé : scenarios, ordres partiels, systèmes distribués, modélisation

Contents

1 Introduction

Scenario languages have met a considerable interest in the last decade. The enthusiasm of the research and engineering community for this kind of notation can be explained by two major characteristics of these languages.

Firstly, from the engineering point of view, scenario languages have a simple and appealing graphical representation. The graphical notation depicts clearly the interaction among processes with only a few concepts: processes, messages, internal actions. Figure 1 shows an example of a simple protocol described with High-level Message Sequence Charts (or HMSCs): two processes p and q exchange messages m and n, and then terminate their interaction with a message o. After sending m, process p systematically performs an internal action a. Scenario languages have been integrated into UML [21], and have become *de facto* a standard notation.

Secondly, from the research community point of view, scenario languages are well formalized: up to some variation, scenario languages can be defined as finite state automata over an alphabet of labeled partial orders. The HMSC of Figure 1 can be seen as an automaton labeled by MSCs $M1$ and $M2$. These automata can be considered as generators of (usually infinite) families of labeled partial orders and have an interesting expressive power. Furthermore, they are a true concurrency model, and as a consequence, one can expect to solve some problems more efficiently with scenario languages than with an equivalent interleaved model. In this report, we will only consider High-level Message Sequence Charts (or HMSCs for short), which are automata over an alphabet of Message Sequence Charts (MSCs), a particular kind of labeled partial orders that are closed for communication (i.e each message sent in a MSC M is also received in M). However, most of results on HMSCs can be easily adapted to other classical scenario languages (Live Sequence charts [12], UML's sequence diagrams [21], ...).

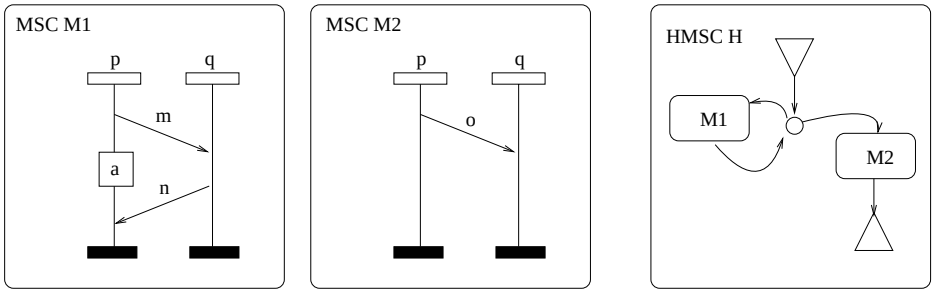

Figure 1: A simple distributed protocol described with 2 MSCs and a HMSC

However, classical scenario languages also have some major drawbacks. First, it has been shown that HMSCs have, at least, the expressive power of semi-traces [20] and Mazurkiewicz

traces [19]. These results indicate that most of usual problems related to specification languages are undecidable for scenario languages. For instance, if we consider two HMSCs H_1 and H_2, and call F_1 and F_2 the sets of partial orders generated by H_1 and H_2 one can not decide whether H_1 and H_2 define a common behavior (i.e. $F_1 \cap F_2 = \emptyset$). One can not either model-check an HMSC against a property expressed in a temporal logic such as CTL. That could have meant that scenario are unusable, at least as a specification language, because they are not amenable to verification.

To tackle this first drawback, several subclasses of scenario languages have been identified: regular High-level MSCs [3], for which the set of all linearizations form a regular language, globally cooperative MSCs [19, 8], for which the common behavior problem described above becomes decidable. Recently, [14] has shown that diagnosis (finding a complete execution of a model from a partial observation) is decidable for HMSCs.

The second drawback of classical scenario languages is the following: when we consider the set of MSCs generated by a HMSC, these MSC languages are unsurprisingly finitely generated, i.e all MSCs in the language can be defined as the sequential composition of MSCs chosen from a finite set of MSCs [18]. However, most protocols used nowadays exhibit behaviors that are not finitely generated, for example, sliding windows scenarios: a protocol may involve a sequence of questions and answers from a process A to a process B. A does not have to wait for the answer to a question before sending up to n several other new questions. The value n is then called the size of the sliding window. Behaviors of this kind of protocols often figure messages from A to B and messages from B to A crossing over the network. They may even resemble an infinite braid as in the example of Figure 2, where a message from A to B systematically crosses at least one message from B to A, and conversely. Clearly, a set of behaviors containing such braids of arbitrary sizes can not be defined by concatenation of elements chosen from a finite set of finite MSCs.

To tackle that second drawback, an immediate solution is to extend scenario languages by allowing a partition of message emissions and receptions in MSCs and match emissions and receptions at the time of composition. This solution was proposed in [11], and is called Compositional Message Sequence Charts (or CMSCs for short). However, CMSCs have the expressive power of communicating automata (which are known to be as powerful as Turing Machines [5]). Consequently, several problems that are decidable for HMSCs become undecidable for CMSCs. For instance, if we consider a HMSC H, it is trivial to know if there exists an MSC generated by H that contains a given message m. This trivial property becomes undecidable for CMSCs. In order to resolve simultaneously decidability and expressiveness drawbacks, several classes of CMSCs [17, 4, 11], sometimes called realizable or safe CMSC, were defined to ensure decidability [10]. Safe CMSCs generate only $\exists$-bounded MSCs, i.e. behaviors that have at least one linearization where communication channels do not exceed a certain bound B.

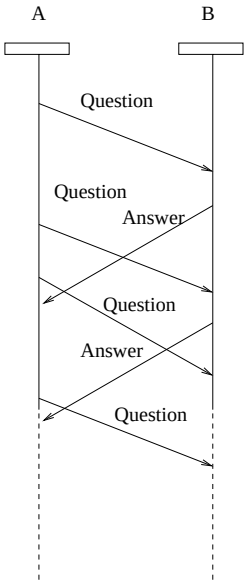

Figure 2: An infinite braid

This paper proposes another approach to extend HMSC expressiveness while keeping the formalism tractable. It consists in a generalization of the sequential composition mechanisms that allows some commutation among composed MSCs. The resulting generalized model is called *causal HMSCs*. It is uncomparable with safe CMSCs, the main reason being that FIFO cannot be ensured in such scenarios. It also implies that the main method used to solve decidability for safe CMSCs - the Kuske's alphabet to encode the problem into traces - cannot be applied since the n-th message sent on a channel is not necessarily the n-th message received on that channel. Instead, we generalize the method of [9] based on atoms, i.e. parts of MSCs that can not be obtained as a concatenation of smaller parts.

This new model is defined in section 2. The following sections study properties of causal HMSCs: section 3 shows the existence of a decidable subclass of causal HMSCs called tight causal HMSCs, where the linearizations of the orders generated form regular languages. Section 4 shows the existence of a decidable subclass called globally-cooperative Causal HMSCs, where the emptiness of the intersection of two sets of orders is decidable. Section 5 shows the existence of a decidable subclass called bounded window causal HMSCs where each message of a generated order is crossed by a bounded number of messages. Section 6 is a case study. A part of the TCP protocol that can not be modeled with HMSCs is designed with causal HMSCs. Section 7 concludes this work.

2 Causal MSCs and Causal HMSCs

Through the rest of the paper, we fix a finite nonempty set $\mathcal{P}$ of process names where $|\mathcal{P}| > 1$, a finite nonempty set *Msg* of message names, *Act* a finite nonempty set of internal action names. We define the alphabets $\Sigma_! = \{p!q(m) \mid p, q \in \mathcal{P}, p \neq q, m \in Msg\}$, $\Sigma_? = \{p?q(m) \mid p, q \in \mathcal{P}, p \neq q, m \in Msg\}$, and $\Sigma_{act} = \{p(a) \mid p \in \mathcal{P}, a \in Act\}$. The letter $p!q(m)$ means the sending of message m from p to q; $p?q(m)$ the reception of message m at p from q; and $p(a)$ the execution of internal action a by process p. Set $\Sigma = \Sigma_! \cup \Sigma_? \cup \Sigma_{act}$. We define the *location* of a letter α in Σ, denoted $loc(\alpha)$, by $loc(p!q(m)) = p = loc(p?q(m)) = loc(p(a))$. For each process p in $\mathcal{P}$, we set $\Sigma_p = \{\alpha \in \Sigma \mid loc(\alpha) = p\}$.

Definition 1 *A* causal MSC *over* $(\mathcal{P}, \Sigma)$ *is a structure* $B = (E, \lambda, \{\sqsubseteq_p\}_{p\in\mathcal{P}}, \ll)$ *where*

- E *is a finite nonempty set. Members of* E *are called* events.
- $\lambda : E \to \Sigma$ *is a labeling function. This labeling function allows for a partition of* E *into three subsets,* $E_! = \lambda^{-1}(\Sigma_!)$,$E_? = \lambda^{-1}(\Sigma_?)$ *and* $E_{act} = \lambda^{-1}(\Sigma_{act})$
- *For each process* p *in* $\mathcal{P}$, $\sqsubseteq_p \subseteq E_p \times E_p$ *is a partial order on* E_p, *where* $E_p = \{e \in E \mid loc(\lambda(e)) = p\}$.
- $\ll \subseteq E_! \times E_?$ *identifies message emission-reception pairs. Thus,* $\ll$ *satisfies the following conditions:*
  - *For each* $e \in E_!$, *there is at most one* $e' \in E_?$ *such that* $(e, e') \in \ll$. *For each* $e \in E_?$, *there is at most one* $e' \in E_!$ *such that* $(e', e) \in \ll$.
  - *For each* $(e, e') \in \ll$, *if* $\lambda(e) = p!q(m)$, *then* $\lambda(e') = q?p(m)$.
- *The relation* $(\bigcup_{p\in\mathcal{P}} \sqsubseteq_p) \cup \ll$ *is acyclic.*

Let $B = (E, \lambda, \{\sqsubseteq_p\}_{p\in\mathcal{P}}, \ll)$ be a causal MSC as above. For convenience, we will often drop the subscript $p \in \mathcal{P}$ and let p range over $\mathcal{P}$. We will also write B interchangeably as $(E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$, where $\leq \subseteq E \times E$ is the transitive closure of $(\bigcup_{p\in\mathcal{P}} \sqsubseteq_p) \cup \ll$. We define $Alph(B) = \{\lambda(e) \mid e \in E\}$. For each p in $\mathcal{P}$, we set $Alph_p(B) = Alph(B) \cap \Sigma_p$. Moreover, we define the *communication graph* of B, denoted by CG_B, to be the directed graph $(Q, \rightsquigarrow)$, where $Q = \{p \in \mathcal{P} \mid E_p \neq \emptyset\}$ and $\rightsquigarrow \subseteq Q \times Q$ is given by: $(p, q) \in \rightsquigarrow$ iff $\ll \cap (E_p \times E_q) \neq \emptyset$.

A *linearization* of B is a sequence $a_1 a_2 \ldots a_\ell$ in Σ^ℓ, where $\ell = |E|$, such that the elements of E can be listed as $e_1, e_2, \ldots, e_\ell$ with $\lambda(e_i) = a_i$ for each i; and $e_i \leq e_j$ implies $i \leq j$ for any i, j. We let $Lin(B)$ denote the set of linearizations of B.

As for MSCs, causal MSCs can be represented graphically. In MSCs, each process is represented as a vertical line (also called lifeline), that symbolizes time elapsing from top to bottom. Along a lifeline, events are totally ordered. For causal MSCs, we will keep the same representation, except that we will represent each process as a labeled partial order

(instead of a labeled total order). Figure 3 depicts a causal MSC M_1. In M_1, events located on process p or on process q are unordered. This is symbolized by two boxes attached to instance names p and q, containing two unordered events, respectively e_1 and e_2 for process p, and f_1 and f_2 for process q. The linearizations defined by this causal MSC are:

$$p!q(m).q?p(m).p!q(n).q?p(n)$$
$$p!q(m).p!q(n).q?p(m).q?p(n)$$
$$p!q(m).p!q(n).q?p(n).q?p(m)$$
$$p!q(n).p!q(m).q?p(n).q?p(m)$$
$$p!q(n).p!q(m).q?p(m).q?p(n)$$
$$p!q(n).q?p(n).p!q(m).q?p(m)$$

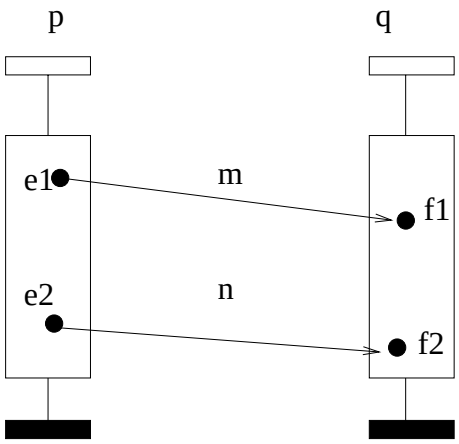

Figure 3: An example of causal MSC M_1. Events are not totally ordered on each processes.

Definition 2 *An* extension *of a causal MSC $B = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$ is a causal MSC $B' = (E', \lambda', \{\sqsubseteq'_p\}, \ll', \leq')$ such that $E' = E$, $\lambda' = \lambda$, for each p, $\sqsubseteq'_p$ is a superset of $\sqsubseteq_p$, and $\ll'=\ll$. Consequently, $\leq'$ is a superset of $\leq$.*

If each $\sqsubseteq'_p$ is a total order, then we call B' a visual extension *of B. For the causal MSC B, we will denote by $Vis(B)$ the set of all its visual extensions.*

Visual extensions of causal MSCs are MSCs, usually called basic Message Sequence Charts in the literature. The initial idea of visual ordering comes from [2], that notices that depending on the interpretation of an MSC, for example when a lifeline describes a physical entity in a network, imposing an ordering on message receptions is not possible. Hence, [2] distinguishes two orderings on MSCs: a visual order, that comes from the relative order of events along an instance line, and a causal order, that is weaker, and does not impose any ordering among consecutive receptions.

Note that the set of visual extensions of a causal MSC B is not necessarily the union of instance per instance linearizations, as an extension of a causal MSC must remain a MSC. For example, for the causal MSC B of Figure 4 in any visual extension $V = (\{e_1, e_2, f_1, f_2\}, \lambda, \{\sqsubseteq_p, \sqsubseteq_q\}, \ll) \in Vis(B)$ we can not have $e_2 \sqsubseteq_p e_1$ and $f_1 \sqsubseteq_q f_2$ at the same time.

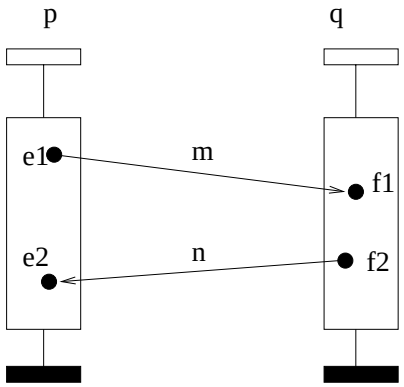

Figure 4: An example of causal MSC B: the set of visual extensions of B is not the instance per instance commutative closure of any visual extension of B.

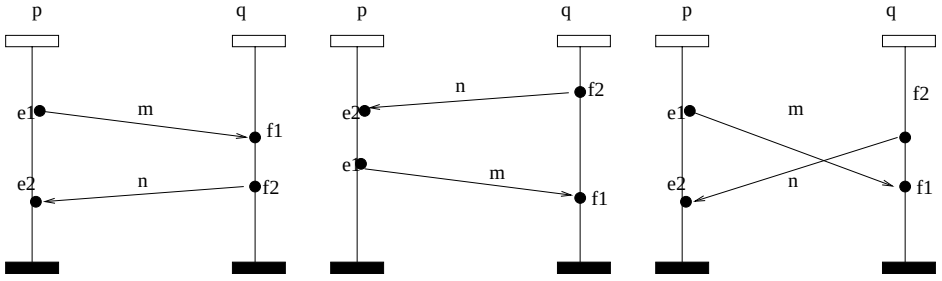

Figure 5: Visual extensions of the causal MSC B of Figure 4

Even with the possibility to define partial orders on each process, Causal MSCs still define finite behaviors. We hence have to define composition operators that allow for the definition of unbounded behaviors. For each process p in $\mathcal{P}$, we fix a concurrent alphabet (Σ_p, I_p), where I_p is an independence relation over the alphabet of actions Σ_p, i.e. $I_p \subseteq \Sigma_p \times \Sigma_p$ is a symmetric and irreflexive relation. We denote the dependence relation $(\Sigma_p \times \Sigma_p) - I_p$ by D_p. For convenience, we also define $I = \bigcup_{p \in \mathcal{P}} I_p$ and $D = \bigcup_{p \in \mathcal{P}} D_p$. Following the usual

definitions of Mazurkiewicz traces, for each trace alphabet (Σ_p, I_p), we define the associated trace equivalence $\sim_p$ over $\Sigma_p^\star$ to be the least equivalence relation such that, for any u, v in $\Sigma_p^\star$ and a, b in Σ_p, $a\ I_p\ b$ implies $uabv \sim_p ubav$. We let $[u]_p$ (or simply $[u]$) denote the equivalence class containing u, with respect to $\sim_p$. We can now define the composition of two causal MSCs.

Definition 3 *Let $B = (E, \lambda, \{\sqsubseteq_p\}, \ll)$ and $B' = (E', \lambda', \{\sqsubseteq'_p\}, \ll')$ be causal MSCs. We define the* concatenation *of B with B', denoted by $B \circledcirc B'$, to be the causal MSC $B'' = (E'', \lambda'', \{\sqsubseteq''_p\}, \ll'')$ where*

- *E'' is the disjoint union of E and E'.*
- *$\lambda''(e) = \lambda(e)$ for $e \in E$ and $\lambda''(e) = \lambda'(e)$ for $e \in E'$.*
- *For each process p in $\mathcal{P}$, $\sqsubseteq''_p$ is the transitive closure of $\sqsubseteq_p \bigcup \sqsubseteq'_p \bigcup \{(e, e') \in E_p \times E'_p \mid \lambda(e)\ D_p\ \lambda'(e')\}$.*
- *$\ll''=\ll \cup \ll'$.*

Clearly $\circledcirc$ is a well-defined and associative operation. We can note that composition of causal MSCs is more general than the usual composition of traces defined as $[u].[v] = [uv]$, as concurrency on processes in each causal MSC need not be compatible with the independence relations $\{I_p\}_{p\in\mathcal{P}}$. Hence we can define a causal MSC such that $e_1 \sqsubseteq_p e_2$ and $\lambda(e_1) I_p \lambda(e_2)$. So, for a causal MSC B we can not in general find a word u such that $Lin(B) = [u]$.

Note also that if B, B' are MSCs and $D_p = \Sigma_p \times \Sigma_p$ for every $p \in \mathcal{P}$, then $B \circledcirc B'$ is equivalent to the usual weak sequential composition of B with B' (denoted $B \circ B'$). We recall that the usual weak sequential composition preserves the ordering in B and B', and impose an ordering between events of B and B' located on the same process. For more information on weak sequencing, and more generally on the semantics of Message Sequence Charts, the interested reader is referred to [22].

As for MSCs, causal MSCs are not expressive enough to define an interesting language. Hence, causal MSCs must be equipped with the usual operations met in process algebras, to allow choices, iterations, etc. As for HMSCs, this will be done using an automaton over an alphabet of causal MSCs, called causal High-level MSC.

Definition 4 *A* causal High-level MSC *("causal HMSC" for short) is a structure $H = (N, N_{in}, \mathcal{B}, N_{fi}, \longrightarrow)$ where:*

- *N is a finite nonempty set. Members of N are called* nodes.
- *$N_{in} \subseteq N$ is the set of initial nodes.*
- *$\mathcal{B}$ is a finite nonempty set of causal MSCs.*
- *$N_{fi} \subseteq N$ is the set of final nodes.*

- $\longrightarrow \subseteq N \times \mathcal{B} \times N$ *is the transition relation.*

Let $H = (N, N_{in}, \mathcal{B}, \longrightarrow, N_{fi})$ be a causal HMSC. A *path* in H is a sequence $\rho = n_0 \xrightarrow{B_1} n_1 \xrightarrow{B_2} n_2 \cdots n_{\ell-1} \xrightarrow{B_\ell} n_\ell$. We refer to n_0 as the starting node and to n_ℓ as the ending node of path ρ. The path ρ is *accepting* iff $n_0 \in N_{in}$ and $n_\ell \in N_{fi}$. A *cycle* in H is a path whose starting and ending nodes are equal. The causal MSC generated by ρ, denoted by $\circledcirc(\rho)$, is $B_1 \circledcirc B_2 \circledcirc \cdots \circledcirc B_\ell$. A causal MSC B is said to be generated by H iff B is generated by an accepting path of H. We let $cMSC(H)$ denote the set of causal MSCs generated by H. Similarly, we will denote by $Vis(H)$ and $Lin(H)$ the set of visual extensions and linearizations of all causal MSCs in $cMSC(H)$. We can here immediately remark that $Lin(H) = \{Lin(v) \mid v \in Vis(H)\}$, i.e. the set of linearizations generated by H is the set of linearizations generated by its visual extensions. Finally, note that we do not require any ordering among receptions of messages, even for messages of the same kind. Hence, messages exchanged between two processes in the same direction are not necessarily delivered at the same speed and can cross each other. This is a major difference with compositional MSCs, where it is crucial to have a FIFO ordering among messages of the same type.

Definition 5 *A MSC language L is* finitely generated *[18] iff there exists a finite set of MSCs X such that any MSC B in L can be defined as the weak sequential composition of elements of X, i.e. $B = B_1 \circ B_2 \cdots \circ B_t$ for some $B_1, \ldots, B_t$ in X.*

By definition, High-level Message Sequence Charts define finitely generated MSC languages. Note however that all finitely generated MSC languages are not HMSC languages: $B_1^n.B_2^n$ is finitely generated but is not an HMSC language. For a causal HMSC H, $Vis(H)$ is not necessarily finitely generated. Consider for instance the causal HMSC H of Figure 6, with the following dependence relation $D = \{(p!q(m), p!q(m)); (p?q(n), p?q(n))\}$. The language of visual extensions of H contains braids interleaving messages m and n an arbitrary number of times, such as the one represented on the right part of the figure. Clearly, the MSC language $Vis(H)$ is not finitely generated.

Let us compare the expressive powers of the different scenario models. Firstly, it is obvious that causal HMSCs embed the expressive power of HMSCs: if we choose as definition of the dependency relation the set of all pairs of events located on similar processes ($\forall p \in \mathcal{P}, D_p = \Sigma_p \times \Sigma_p$), then the set of visual extensions generated by a causal HMSC with such dependency relation is the set of MSCs generated by a HMSC.

Second, causal HMSCs and CMSCs are uncomparable. Consider, for example the safe CMSC of Figure 7. In CMSCs, a messages need not be complete in the alphabet labeling the partial order automaton. Pending emissions and receptions are associated during concatenation to form messages, and the language generated by a CMSC is the set of MSCs obtained this way. To pair message emissions and receptions, the i^{th} emission of a message of type m is associated to the i^{th} reception of the same type of message. Hence, two occurrences of a given message type can not overtake. The CMSC depicted in the left part of Figure 7 defines a set of executions where an arbitrary number of occurrences of message m cross a single

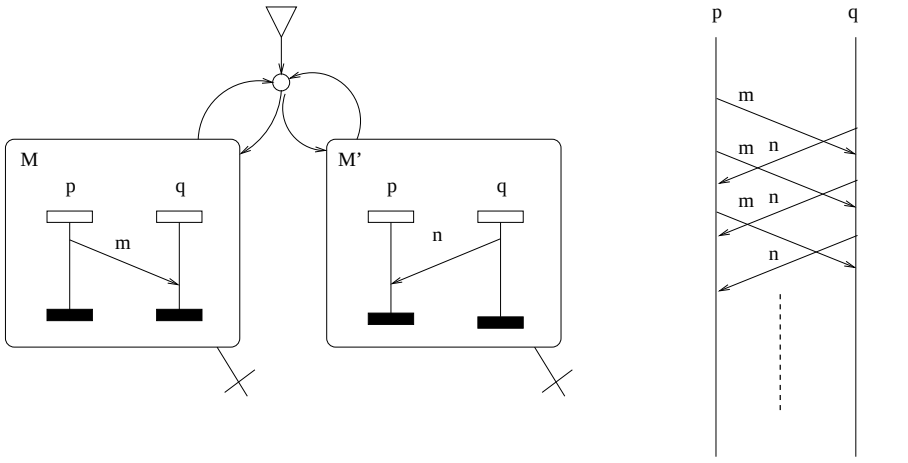

Figure 6: An example of Causal HMSC H, with $D = \{(p!q(m), p!q(m)); (p?q(n), p?q(n))\}$, and a behavior in $Vis(H)$

occurrence of message n. This set of behaviors can not be represented as a causal HMSC: to allow executions where message n crosses an arbitrary number of occurrences of message m, causal MSCs should allow commutations between m and n, and hence would also define behaviors where m is sent arbitrarily often before n. The causal HMSC depicted on the right of the figure defines a superset of these executions. Conversely, executions generated by a CMSC need to respect the FIFO ordering among messages of the same kind, which is not required in causal HMSCs. Hence Causal MSCs and CMSCs are uncomparable.

Note also that causal HMSCs allow to answer some questions that are undecidable for Communicating automata (or equivalently CMSCs): let us consider a message m from a process p to a process q. Let us try to answer the following question: is there a MSC generated by H where message m is sent and received? If $H = (N, N_{in}, \mathcal{B}, \longrightarrow, N_{fi})$ is a HMSC or a causal HMSC, answering this question is trivial, and just consists in checking whether there exists an MSC or a causal MSC in $\mathcal{B}$ that contains the message m. For communicating automata, this question is known to be undecidable [11].

3 Linearization Languages of Causal HMSCs

An embedding of Mazurkiewicz traces into HMSCs is described in [20, 19]. Following basic results in Mazurkiewicz trace theory [7], this embedding means that it is undecidable to determine whether the linearization language of a given HMSC is regular. In [3, 19], a subclass of HMSCs, called bounded HMSCs, was identified such that the linearization language of every bounded HMSC is regular. As already mentioned, causal HMSCs embed

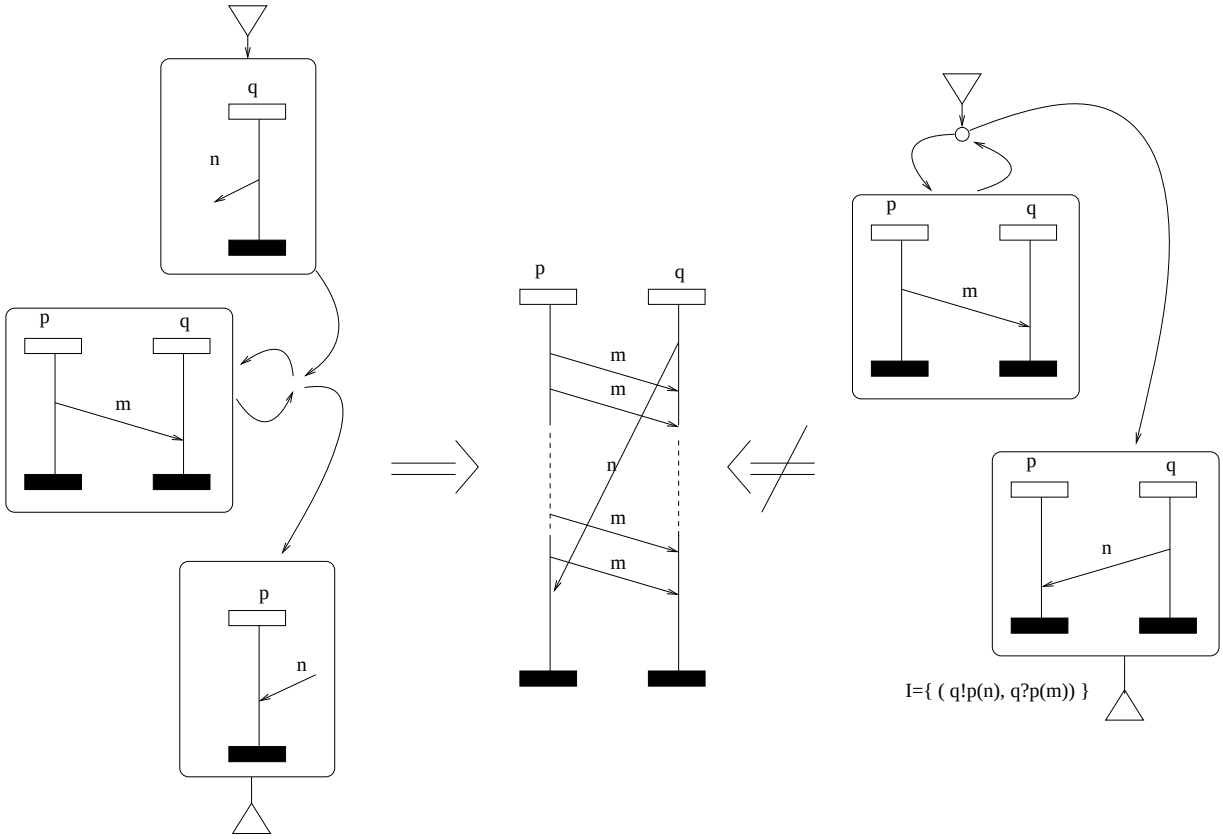

Figure 7: Comparing CMSCs and Causal HMSCs

the expressive power of HMSCs. Thus, due to [3, 19] it is undecidable to test whether the linearization language of a given causal HMSC is regular. We show hereafter a sufficient condition for a causal HMSC to have a regular linearization language.

We recall the notion of connectedness from Mazurkiewicz trace theory ([7]). Let $p \in \mathcal{P}$. We say $\Gamma \subseteq \Sigma_p$ is D_p-connected iff the (undirected) graph $(\Gamma, D_p \cap (\Gamma \times \Gamma))$ is connected. Now we say the causal MSC B is *tight* iff its communication graph CG_B is strongly connected and moreover for every $p \in \mathcal{P}$, $Alph_p(B)$ is D_p-connected. If however the communication graph CG_B is weakly connected and moreover for every $p \in \mathcal{P}$, $Alph_p(B)$ is D_p-connected, then we will say that B is *weakly tight*.

Let $H = (N, N_{in}, \mathcal{B}, N_{fi}, \longrightarrow)$ be a causal HMSC. We say that H is *tight* iff for every cycle ρ in H, the causal MSC $\circledcirc(\rho)$ is tight. Equivalently, H is tight iff for every strongly connected subgraph G of H such that $\{B_1, \ldots, B_\ell\}$ is the set of causal MSCs appearing in G, $B_1 \circledcirc \ldots \circledcirc B_\ell$ is tight. Note that tightness of $B_1 \circledcirc \ldots \circledcirc B_\ell$ does not depend on the order in which $B_1, \ldots, B_\ell$ are listed.

Following these definitions, we can now establish the main result of this section:

Theorem 1 *Let $H = (N, N_{in}, \mathcal{B}, N_{fi}, \longrightarrow)$ be a tight causal HMSC. Then $Lin(H)$ is a regular subset of $\Sigma^\star$ i.e. we can build an automaton over Σ such that it recognizes $Lin(H)$.*

Notice that a weakly thight (causal) HMSC may not be bounded, hence $Lin(H)$ may not be regular. For instance, consider a causal HMSC H that iterates a single message, i.e. H is a loop labeled by a MSC that contains a message (a, b), with $a = p!q$ and $b = q?p$. $Lin(H)$ is a Dyck or parenthesis language, which is well known not to be regular. Surprisingly, we do not need $Lin(H)$ to be regular to be able to model check HMSCs, as shown in [9] and also in the next section.

The result for (non causal) tight HMSC was proved by [16] using an encoding into connected traces and building an automaton which recognizes such connected traces. In our case, finding such embedding into Mazurkiewicz traces seems impossible due to the non Fifoness. Thus, we adapt the proof of regularity of trace closures of loop-connected automata from [7, 19].

The rest of this section is now devoted to the proof of Theorem 1. We fix a tight causal HMSC H as in the theorem, and show that we can build a finite state automaton $\mathcal{A}_H$ over Σ which accepts $Lin(H)$.

First, we establish some technical results.

Proposition 1 *Let $\rho = \theta_1 \ldots \theta_2 \ldots \theta_{|\Sigma|}$ be a path of H, where for each $i = 1, \ldots, |\Sigma|$, the subpath $\theta_i = n_{i,0} \xrightarrow{B_{i,1}} n_{i,1} \ldots n_{i,\ell_i - 1} \xrightarrow{B_{i,\ell_i}} n_{i,0}$ is a cycle (these cycles need not be contiguous). Suppose further that the sets $\widehat{\mathcal{B}}_i = \{B_{i,1}, \ldots, B_{i,\ell_i}\}$, $i = 1, \ldots, |\Sigma|$, are equal. Let e be an event in $\circledcirc(\theta_1)$ and e' an event in $\circledcirc(\theta_{|\Sigma|})$. Let $\circledcirc(\rho) = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$. Then we have $e \leq e'$.*

Proof: We consider two cases.

—**Case** (i): $loc(\lambda(e)) = loc(\lambda(e'))$.

Let $p = loc(\lambda(e))$. Let us recall that H is tight, thus $Alph_p(\circledcirc(\theta_1)) = \ldots = Alph_p(\circledcirc(\theta_K))$ is D_p-connected. Consequently, we can find a set of events $\{e_j\}_{j=1,\ldots,t}$, where $t \leq |\Sigma_p| - 2$, each e_j is in $\circledcirc(\theta_{j+1})$, and such that $\lambda(e)\ D_p\ \lambda(e_1)\ D_p\ \ldots\ D_p\ \lambda(e_t)\ D_p\ \lambda(e')$. Thus $e \leq e_1 \leq \ldots \leq e_t \leq e'$.

—**Case** (ii): $loc(\lambda(e)) \neq loc(\lambda(e'))$. Let $p_1 p_2 \ldots p_t$ be a path from $loc(\lambda(e))$ to $loc(\lambda(e'))$ in the communication graph of $\circledcirc(\rho)$, where $p_1 = loc(\lambda(e))$, $p_t = loc(\lambda(e'))$. In view of the arguments in Case (i), it is easy to see that we can pick events e_i, f_i in $\circledcirc(\theta_{u_i})$, $i = 1, \ldots, t-1$, where for each i, $u_i = |\Sigma_{p_1}| + |\Sigma_{p_2}| + \ldots + |\Sigma_{p_i}|$, $loc(e_i) = p_i$, $loc(f_i) = p_{i+1}$ and $e_i \ll f_i$. Thus $e \leq e_1 \ll f_1 \leq e_2 \ll f_2 \leq \ldots \leq e_{t-1} \ll f_{t-1} \leq e'$. □

Let $\rho = n_0 \xrightarrow{B_1} \cdots \xrightarrow{B_\ell} n_\ell$ be a path in H, where $B_i = (E_i, \lambda_i, \{\sqsubseteq_p^i\}, \ll_i)$ for $i = 1, \ldots, \ell$. Let $\circledcirc(\rho) = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$. A *configuration* of ρ is a $\leq$-closed subset of E. Let C be a configuration of ρ. A *C-subpath* of ρ is a maximal subpath $\varrho = n_u \xrightarrow{B_{u+1}} \ldots \xrightarrow{B_{u'}} n_{u'}$, such that $C \cap E_i \neq \emptyset$ for each $i = u, \ldots, u'$. For such a *C-subpath* ϱ, we define its *C-residue* to be the set $(E_{u+1} \cup E_{u+2} \cup \cdots \cup E_{u'}) - C$. Figure 8 illustrates these notions. Each causal MSC is represented by a rectangle. Events in the configuration C are indicated by small filled circles, while events not in C are indicated by small blank circles. The two *C-subpaths* identified on Figure 8 are the sequences of transitions that provide the events appearing in C.

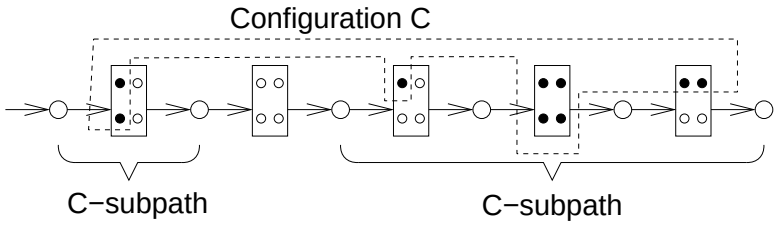

Figure 8: Events in C-subpaths are indicated by small filled circles. Events in C-residues are indicated by small blank circles.

Lemma 1 *Let ρ be a path in H and C be a configuration of ρ. Then,*

(i) The number of C-subpaths of ρ is at most $K_{subpath} = |N| \cdot |\Sigma| \cdot 2^{|\mathcal{B}|}$.

(ii) Let ϱ be a C-subpath of ρ. Then the number of events in the C-residue of ϱ is at most $K_{residue} = |N| \cdot |\Sigma| \cdot 2^{|\mathcal{B}|} \cdot \max\{|B| \mid B \in \mathcal{B}\}$, where $|B|$ denotes the number of events in the causal MSC B.

Proof:

(i) Suppose the contrary. Let $K = |\Sigma| \cdot 2^{|\mathcal{B}|}$. We can find $K+1$ C-subpaths whose ending nodes are equal. Let the indices of these $K+1$ ending nodes be $i_1 < i_2 < \ldots < i_{K+1}$. For $h = 1, \ldots, K$, let θ_h be the subpath of ρ from n_{i_h} to $n_{i_{h+1}}$; and let $\widehat{\mathcal{B}}_h$ be the set of causal MSCs appearing in θ_h. Hence we can find $\theta_{j_1}, \theta_{j_2}, \ldots, \theta_{j_{|\Sigma|}}$, $j_1 < j_2 < \ldots < j_{|\Sigma|}$, such that $\widehat{\mathcal{B}}_{j_1} = \widehat{\mathcal{B}}_{j_2} = \ldots = \widehat{\mathcal{B}}_{j_{|\Sigma|}}$. Pick an event e from $\circledcirc(\theta_{j_1})$ with $e \notin C$. Such an e exists, since, for example, none of the events in the first causal MSC appearing in θ_{j_1} is in C. Pick an event e' from $\circledcirc(\theta_{j_{|\Sigma|}})$ with $e' \in C$. Applying Proposition 1 yields that $e < e'$. This leads to a contradiction, since C is $\leq$-closed.

(ii) Let $\varrho = n_i \xrightarrow{B_{i+1}} \ldots \xrightarrow{B_{i'}} n_{i'}$. Let $\widehat{E}_j = E_j - C$ for $j = i+1, \ldots, i'$. By similar arguments as in (i), it is easy to show that among $\widehat{E}_{i+1}$, $\ldots$, $\widehat{E}_{i'}$, at most $|N| \cdot |\Sigma| \cdot 2^{|\mathcal{B}|}$ of them are nonempty. The claim then follows.

□

We are now ready to define the finite state automaton $\mathcal{A}_H = (S, S_{in}, \Sigma, S_{fi}, \Longrightarrow)$ which accepts $Lin(H)$. As usual, S will be the set of states, $S_{in} \subseteq S$ the initial states, $\Longrightarrow \subseteq S \times \Sigma \times S$ the transition relation, and $S_{fi} \subseteq S$ the final states. Fix $K_{subpath}$, $K_{residue}$ to be the constants defined in Lemma 1. If $B = (E, \lambda, \{\sqsubseteq_p\}, \ll)$ is a causal MSC and E' a subset of E, then we define the restriction of B to E' to be the causal MSC $B' = (E', \lambda', \{\sqsubseteq'_p\}, \ll')$ as follows. As expected, λ' is the restriction of λ to E'; for each p, $\sqsubseteq'_p$ is the restriction of $\sqsubseteq_p$ to $(E' \cap E_p) \times (E' \cap E_p)$; and $\ll'$ is the restriction of $\ll$ to E'.

Intuitively, for a word σ in $\Sigma^\star$, $\mathcal{A}_H$ guesses an accepting path ρ of H and checks whether σ is in $Lin(\circledcirc(\rho))$. After reading a prefix σ' of σ, $\mathcal{A}_H$ memorizes a sequence of subpaths from which σ' was "linearized" (i.e the C-subpath of a path ρ such that C is a configuration reached after playing σ' and $\circledcirc(\rho)$ contains C). With Lemma 1, it will become clear later that at any time, we should remember at most $K_{subpath}$ such subpaths. Moreover, for each subpath, we need to know only a *bounded* amount of information, which will be stored in a data structure called "segment".

A causal MSC $B_i = (E_i, \lambda, \{\sqsubseteq_p\}, \ll)$ is K-bounded if $|E| \leq K$. A *segment* is a tuple (n, Γ, W, n'), where $n, n' \in N$, Γ is a nonempty subset of Σ, and W is either a non-empty $K_{residue}$-bounded causal MSC, or the special symbol $\perp$. The state set S of $\mathcal{A}_H$ is the collection of finite sequences $\theta_1\theta_2\ldots\theta_\ell$, $0 \leq \ell \leq K_{subpath}$, where each θ_i is a segment. Intuitively, a segment (n, Γ, W, n') keeps track of a subpath ϱ of H which starts at n and ends at n'. Γ is the collection of letters of events in $\circledcirc(\varrho)$ that have been "linearized". Finally, W is the restriction of $\circledcirc(\varrho)$ to the set of events in $\circledcirc(\varrho)$ that are not yet linearized. In case all events in $\circledcirc(\varrho)$ have been linearized, we set $W = \perp$. For convenience, we extend the operator $\circledcirc$ by: $W \circledcirc \perp = \perp \circledcirc W = W$ for any causal MSC W; and $\perp \circledcirc \perp = \perp$.

We define $\mathcal{A}_H = (S, S_{in}, \Sigma, S_{fi}, \Longrightarrow)$ as follows:

- As mentioned above, S is the collection of finite sequence of at most $K_{subpath}$ segments.
- The initial state set is $S_{in} = \{\varepsilon\}$, where ε is the null sequence.

- A state is final iff it consists of a single segment $\theta = (n, \Gamma, \perp, n')$ such that $n \in N_{in}$ and $n' \in N_{fi}$ (and Γ is any nonempty subset of Σ).
- The transition relation $\Longrightarrow$ of $\mathcal{A}_H$ is the least set satisfying the following conditions.

  —**Condition** (i):

  Suppose $n \xrightarrow{B} n'$ where $B = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$. Let e be a minimal event in B (with respect to $\leq$) and let $a = \lambda(e)$. Let $\theta = (n, \Gamma, W, n')$ where $\Gamma = \{a\}$. Let $R = E - \{e\}$. If R is nonempty, then W is the restriction of B to R; otherwise we set $W = \perp$. Suppose $s = \theta_1 \ldots \theta_k \theta_{k+1} \ldots \theta_\ell$ is a state in S where $\theta_i = (n_i, \Gamma_i, W_i, n_i')$ for each i. Suppose, for every $e' \in E$ with $e \leq e'$, it is the case that $\lambda(e')\ I_p\ \gamma$ for any $\gamma \in \Sigma_p \cap (\bigcup_{k+1 \leq i \leq \ell} \Gamma_i)$, where $p = loc(e')$.

  - ("create a new segment") Let $\hat{s} = \theta_1 \ldots \theta_k \theta \theta_{k+1} \ldots \theta_\ell$. If $\hat{s}$ is in S, then $s \overset{a}{\Longrightarrow} \hat{s}$. In particular, for the initial state ε, we have $\varepsilon \overset{a}{\Longrightarrow} \theta$.
  - ("add to the beginning of a segment") Suppose $n' = n_{k+1}$. Let $\hat{\theta} = (n, \Gamma \cup \Gamma_{k+1}, \widehat{W}, n'_{k+1})$, where $\widehat{W} = W \circledcirc W_{k+1}$. Let $\hat{s} = \theta_1 \ldots \theta_k \hat{\theta} \theta_{k+2} \ldots \theta_\ell$. If $\hat{s}$ is in S, then $s \overset{a}{\Longrightarrow} \hat{s}$.
  - ("append to the end of a segment") Suppose $n = n'_k$. Let $\hat{\theta} = (n_k, \Gamma_k \cup \Gamma, \widehat{W}, n')$, where $\widehat{W} = W_k \circledcirc W$. Let $\hat{s} = \theta_1 \ldots \theta_{k-1} \hat{\theta} \theta_{k+1} \ldots \theta_\ell$. If $\hat{s}$ is in S, then $s \overset{a}{\Longrightarrow} \hat{s}$.
  - ("glue two segments") Suppose $n = n'_k$ and $n' = n_{k+1}$. Let $\hat{\theta} = (n_k, \Gamma_k \cup \Gamma \cup \Gamma_{k+1}, \widehat{W}, n'_{k+1})$, where $\widehat{W} = W_k \circledcirc W \circledcirc W_{k+1}$. Let $\hat{s} = \theta_1 \ldots \theta_{k-1} \hat{\theta} \theta_{k+2} \ldots \theta_\ell$. If $\hat{s}$ is in S, then $s \overset{a}{\Longrightarrow} \hat{s}$.

  —**Condition** (ii):

  Suppose $s = \theta_1 \ldots \theta_k \theta_{k+1} \ldots \theta_\ell$ is a state in S where $\theta_i = (n_i, \Gamma_i, W_i, n_i')$ for $i = 1, 2, \ldots, \ell$. Suppose $W_k \neq \perp$. Let $W_k = (R_k, \lambda_k, \{\sqsubseteq_p^k\}, \ll_k, \leq_k)$. Let e be a minimal event in W_k and $a = \eta_k(e)$. Suppose, for every $e' \in R_k$ with $e \leq e'$, it is the case that $\eta_k(e')\ I_p\ \gamma$ for any $\gamma \in \Sigma_p \cap (\bigcup_{k+1 \leq i \leq \ell} \Gamma_i)$, where $p = loc(e')$. Let $\hat{\theta} = (n_k, \Gamma_k \cup \{a\}, \widehat{W}, n'_k)$, where $\widehat{W}$ is as follows: Let $\widehat{R} = R_k - \{e\}$. If $\widehat{R}$ is nonempty, then $\widehat{W}$ is the restriction of W to $\widehat{R}$; otherwise $\widehat{W} = \perp$. Let $\hat{s} = \theta_1 \ldots \theta_{k-1} \hat{\theta} \theta_{k+1} \ldots \theta_\ell$. Then we have $s \overset{a}{\Longrightarrow} \hat{s}$. (Note that $\hat{s}$ is guaranteed to be in S.)

We have now completed the construction of $\mathcal{A}_H$. It remains to prove the following lemma to complete the proof of theorem 1:

Lemma 2 *Let $\sigma \in \Sigma^\star$. Then σ is accepted by $\mathcal{A}_H$ iff σ is in $Lin(H)$.*

Proof: Let $\sigma = a_1 a_2 \ldots a_k$. Suppose σ is in $Lin(H)$. Let $\rho = n_0 \xrightarrow{B_1} \ldots \xrightarrow{B_\ell} n_\ell$ be an accepting path in H such that σ is a linearization of $\circledcirc(\rho)$. Hence we may suppose that

$\circledcirc(\rho) = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$ where $E = \{e_1, e_2, \ldots, e_k\}$ and $\lambda(e_i) = a_i$ for $i = 1, \ldots, k$. And $e_i \leq e_j$ implies $i \leq j$ for any i, j in $\{1, \ldots, k\}$. Consider the configurations $C_i = \{e_1, e_2, \ldots, e_i\}$ for $i = 1, \ldots, k$. For each C_i, we can associate a state s_i in $\mathcal{A}_H$ as follows. Consider a fixed C_i. Let $\rho = \ldots \varrho_1 \ldots \varrho_2 \ldots \varrho_h \ldots$ where ϱ_1, ϱ_2, ..., ϱ_h are the C_i-subpaths of ρ. Then we set $s_i = \theta_1 \ldots \theta_h$ where $\theta_j = (n_j, \Gamma_j, W_j, n'_j)$ with n_j being the starting node of ϱ_j, and Γ_j the collection of all $\lambda(e)$ for all events e that are in both $\circledcirc(\varrho_j)$ and C_i. Let R_j be the C_i-residue of ϱ_j. If R_j is nonempty, W_j is the causal MSC $(R_j, \lambda_j, \{\sqsubseteq_p^j\}, \ll_j, \leq_j)$ where λ_j is the restriction of λ to R_j; $\sqsubseteq_p^j$ is the restriction of $\sqsubseteq_p$ to those events in R_j that belong to process p, for each p; and $\ll_j$ the restriction of $\ll$ to R_j. If R_j is empty, then set $W_j = \perp$. Finally, n'_j is the ending node of ϱ_j.

Now it is routine (though tedious) to verify that $\varepsilon \stackrel{a_1}{\Longrightarrow} s_1 \ldots s_{k-1} \stackrel{a_k}{\Longrightarrow} s_k$ is an accepting run of $\mathcal{A}_H$. Conversely, given an accepting run of $\mathcal{A}_H$ over σ, it is straightforward to build a corresponding accepting path of H.

□

We have then shown that tightness allows the construction of an automaton $\mathcal{A}_H$ that recognizes all linearizations of H.□

As for complexity, it is easy to see that the number of states of $\mathcal{A}_H$ is at most $\left(|N|^2 \cdot 2^{|\Sigma|} \cdot 2^{|N| \cdot |\Sigma| \cdot 2^{|\mathcal{B}|} \cdot m}\right)^{|N| \cdot |\Sigma| \cdot 2^{|\mathcal{B}|}}$, where $m = \max\{|B| \mid B \in \mathcal{B}\}$.

4 Inclusion and Intersection of Causal HMSCs

As already mentioned, due to the embedding of Mazurkiewicz traces into Message Sequence Charts, we can not decide whether the intersection of two MSC languages generated by HMSCs is empty or not, or if a MSC language is included in an other one. Fortunately, for a subclass of HMSCS called globally-cooperative HMSCs, and which captures in particular bounded HMSCs, this problem becomes decidable [9]. This section shows that a similar class exists for Causal HMSCs.

This class of globally cooperative causal HMSCs heavily depends on how causal MSCs generated by the order automaton can be decomposed into elementary parts. Such decomposition into *atomic MSCs* was proposed for MSCs in [1, 13]. As for MSC, we can introduce a notion of decomposition into basic parts for causal MSCs.

Definition 6 *A causal MSC B is a* basic part *(w.r.t. a dependency relation D) if it can not be decomposed into two (non-empty) causal MSCs B_1 and B_2 such that $B = B_1 \circledcirc B_2$. Basic parts are units of composition, they will be useful to find a canonical representation of any causal MSC.*

For a Causal MSC B a decomposition *is a sequence of basic parts $B_1.B_2 \ldots B_k$ such that $B = B_1 \circledcirc B_2 \circledcirc \ldots \circledcirc B_k$.*

Slightly extending the independence relation I_p defined in section 2, we will say that two sets A and B are independent, denoted by $A\ I_p B$ if for all a_i in A and b_i in B,

$a_i\, I_p\, b_i$. Similarly, we will say that two basic parts B_1 and B_2 are independent, denoted by $B_1 \parallel B_2$, if their labels are instance-wise mutually independent, i.e. for all process p in $\mathcal{P}$, $Alph_p(B_1) I_p Alph_p(B_2)$. Note that whenever $B_1 \parallel B_2$, we have $B_1 \circledcirc B_2 = B_2 \circledcirc B_1$.

This property allows us to define an associated *trace equivalence* as the least congruence $\sim$ over $\mathcal{B}^\star_{\text{Basic}}$, where $\mathcal{B}_{\text{Basic}}$ is a given collection of basic parts, such that $\forall a, b \in \mathcal{B}_{\text{Basic}}$, $a \parallel b \Longrightarrow ab \sim ba$. Thus, we can consider trace $[u]$ over basic parts as the equivalence class of any word of basic parts $u \in \mathcal{B}^\star_{\text{Basic}}$.

Moreover, given a causal HMSC H, we will denote by $\mathcal{B}_{\text{Basic}}(H)$ the finite set of basic parts occurring in the decomposition of all causal MSCs which label transitions of H. We also define H_{Basic} as a new causal HMSC in which we replace each transition (n_{src}, B, n_{dst}) by nodes and transitions $(n_{src}, B_1, n_1) \ldots (n_{k-1}, B_k, n_{dst})$ where $B_1, \ldots, B_k$ is a decomposition of B, and $(n_i)_{i \in \{1..k\}}$ are fresh states. $\mathcal{L}(H_{\text{Basic}}) \subseteq \mathcal{B}^\star_{\text{Basic}(H)}$ is the language over $\mathcal{B}_{\text{Basic}}(H)$ generated by H_{Basic}, i.e. a word $w = B_1.B_2.\ldots.B_k$ belongs to $\mathcal{L}(H_{\text{Basic}})$ if and only if there exists an accepting path $\rho = n_0 \xrightarrow{B_1} n_1 \ldots \xrightarrow{B_k} n_k$ in H_{Basic}. Finally, we will denote by $BP(H) = \{B_1 \ldots B_k \in \mathcal{B}^\star_{\text{Basic}(H)} \mid B_1 \circledcirc \ldots \circledcirc B_k \in cMSC(H)\}$ the language of basic parts generated by H.

We can now consider the trace language generated by a causal HMSC H. We will identify a subclass of causal HMSCs, called globally-cooperative causal HMSCs, for which each of its member H corresponds to a connected rational trace language over the basic part alphabet $\mathcal{B}_{\text{Basic}}(H)$.

A brute force solution to find a decomposition of a causal MSC $B = (E, \lambda, \{\sqsubset_p\}_{p \in \mathcal{P}}, \ll)$ is to check for all possible partitions of E, whether the reconstruction of the decomposition is exactly B. The following proposition shows that decomposition construction can be brought back to the detection of strongly connected components in a directed graph defined over events of E. This construction is analogous to the technique in [13], and can be performed in $O(|E|^2)$.

Proposition 2 *Let us note $\preccurlyeq_p$ the cover relation of $\sqsubseteq_p$, i.e. $e \preccurlyeq_p e'$ iff $e \sqsubseteq_p e'$ and no e_i is such that $e \sqsubseteq_p e_i \sqsubseteq_p e'$. These two problems are equivalent:*

1. *$B = (E, \lambda, \sqsubseteq_p, \ll)$ can be decomposed into $B_1 \ldots B_k$, where $B_i = (E_i, \lambda_i, \sqsubseteq_p^i, \ll_i)$ is a basic part.*
2. *The graph $\mathcal{G}_B = (E, R)$ has $\{E_{B_i}\}_{i \in [1..k]}$ as strongly connected components and $\forall i < j, E_{B_j} \times E_{B_i} \cap R = \emptyset$. A pair of events $(e, e') \in E$ belongs to R if and only if:*
   - *they are locally related in B, i.e. $\exists p \in \mathcal{P} \mid e \preccurlyeq_p e'$,*
   - *or they are reception and emission of the same message, i.e. $e \ll e'$ or, $e' \ll e$*
   - *or they are not compatible with local independency relation, i.e. $\exists p \in \mathcal{P} \mid e' \preccurlyeq_p e$ and $\lambda(e) I_p \lambda(e')$,*
   - *or they are not compatible with local dependency relation, i.e. $\exists p \in \mathcal{P} \mid e' \not\sqsubseteq_p e$, $e \not\sqsubseteq_p e'$, and $\lambda(e) D_p \lambda(e')$.*

Proof: Let us note $R^{-1} = \{(e,e') \mid (e',e) \in R\}$, $R_{I_p} = \{(e',e) \in E \times E \mid e \preccurlyeq_p e' \wedge \lambda(e) I_p \lambda(e')\}$ and $R_{D_p} = \{(e,e') \in E \times E \mid e \not\sqsubseteq_p e' \wedge e' \not\sqsubseteq_p e \wedge \lambda(e) D_p \lambda(e')\}$. Thus, we can rewrite R as $R = \ll \cup \ll^{-1} \cup (\preccurlyeq_p \cup R_{I_p} \cup R_{D_p})_{p \in \mathcal{P}}$.

First, let us take two causal MSCs $X_1 = (E_1, \lambda_1, \sqsubseteq_p^1, \ll_1)$ and $X_2 = (E_2, \lambda_2, \sqsubseteq_p^2, \ll_2)$ and consider the causal MSC $X = X_1 \circledcirc X_2$. We can remark that we have $R_{I_p} \cap (E_2 \times E_1) = \emptyset$ and $R_{D_p} \cap (E_2 \times E_1) = \emptyset$. Thus, $R \cap (E_2 \times E_1) = \emptyset$, and consequently E_X is not a strongly connected component.

Second, let us prove that if E_X is not a strongly connected component of $\mathcal{G}_B$ then X is not a basic part of B. If E_X is not strongly connected component, that means that we can decompose it into n strongly components $E_1 \dots E_n$, with $n > 1$. We can then extend R to these subsets $(E_i, E_j) \in R$ if $\exists e_i \in E_i, e_j \in E_j$ such that $(e_i, e_j) \in R$. We can be more precise: $(E_i, E_j) \in R$ iff $\exists e_i \in E_i, e_j \in E_j$ such that $(e_i, e_j) \in R \backslash R^{-1}$. Indeed, if (e_i, e_j) is in R^{-1} then E_i is strongly connected to E_j which is not the case as they are distinct sets. Moreover, as $\ll$ and R_{D_p} are reflexive, and $R_{I_p} \cap (R_{D_p} \cup R_{D_p}^{-1}) = \emptyset$, $R_{I_p} \cap (\ll \cup \ll^{-1}) = \emptyset$, $\preccurlyeq_p \cap (R_{D_p} \cup R_{D_p}^{-1}) = \emptyset$ and $R_{I_p} \subseteq \preccurlyeq_p^{-1}$, we have $R \backslash R^{-1} = \cup_{p \in \mathcal{P}} (\preccurlyeq_p \backslash R_{I_p}^{-1}) = \{(e,e') \mid \exists p \in \mathcal{P},\ e \preccurlyeq_p e' \wedge \lambda(e) D_p \lambda(e')\}$. Thus, $(E_i, E_j) \in R$ iff for all $e_i \in E_i$, $e_j \in E_j$, $e_i \preccurlyeq_p e_j \Rightarrow \lambda(e_i) D_p \lambda(e_j)$. Finally, we deduce from the later the following result : $R \cap (\cup_{p \in \mathcal{P}} \preccurlyeq_p) \cap (E_i \times E_j) = \{(e_i, e_j) \in E_i \times E_j \mid \lambda(e_i) D_p \lambda(e_j)\}$, which means that $X_{E_i} \circledcirc X_{E_j} = X_{E_i \cup E_j}$ (X_E is the causal MSC resulting of the projection of causal MSC X to events of E). To conclude the proof, we have to consider a linearization $E_{i_1} \dots E_{i_n}$ of the n strongly components, and then the composition $X_{E_{i_1}} \circledcirc X_{E_{i_2} \cup \dots \cup E_{i_n}}$ which is equals to X, which mean that X is not a basic part.

□

This proposition gives us an algorithm to compute a possible decomposition of a causal MSC B: we first compute the strongly connected components $B_1, \dots B_k$ of the graph defined above, and then find a linearization $B_{i1} \dots B_{ik}$, that is chosen according to the respective dependencies among basic parts in B. We now prove that all decomposition of the initial causal MSC are equivalent up-to permutations allowed by $||$.

Proposition 3 *Let B be a causal MSC, and $B_1 \dots . B_k$ be a possible decomposition of B. Then any other decomposition of B belongs to $[B_1 \dots B_k]$.*

Proof: Let us suppose that there exists a word of basic parts $W = w_1 . w_2 \dots w_q$ such that $B = w_1 \circledcirc w_2 \circledcirc \dots \circledcirc w_q$ but $W \notin [B_1 \dots B_k]$. We know that $\{B1, \dots B_k\}$ is the finest partition w.r.t. relation R defined in proposition 2. That is, $\{w_1, w_q\} = \{B1, \dots B_k\}$. Hence, we can find a mapping $f : \{w_1, w_q\} \longrightarrow \{B1, \dots B_k\}$ that associates an isomorphic basic part of $\{B1, \dots B_k\}$ to each w_i. Furthermore, two isomorphic basic parts are necessarily ordered. If $W \notin [B_1 \dots B_k]$, then there are two basic parts of W, w_i w_j such that w_i precedes w_j in W, and for any mapping f, $f(w_j)$ precedes $f(w_i)$ in $B_1 \dots B_k$ and $f(w_j)$ and $f(w_i)$ do not commute in $B_1 \dots B_k$. This can only happen if there exists two events $e \in f(w_j)$ and $e' \in f(w_i)$ such that $loc(e) = loc(e')$ and $e \leq e'$ in B. Hence, we can not have $w_i || w_j$ nor $w_i \leq w_j$ in W. Contradiction.

□

We are now ready to prove the main result of this section. The following theorem shows that a causal HMSC defines a rational (Mazurkiewicz) trace language over basic parts.

Theorem 2 *Let H be a causal HMSC, and let $[L]$ be the closure of words of $L \subseteq \mathcal{B}^{\star}_{Basic(H)}$ by the independence relation $\|$. Then $BP(H) = [\mathcal{L}(H_{Basic})]$.*

Proof: First, let us take a word w in $[\mathcal{L}(H_{Basic}]$. Thus $w = B_1 \dots B_k \sim B_{i_1} \dots B_{i_k}$ such that $B_{i_1} \dots B_{i_k} \in \mathcal{L}(H_{Basic})$. As $\mathcal{L}(H_{Basic}) \subseteq BP(H)$ and $B_1 \circledcirc \dots \circledcirc B_k = B_{i_1} \circledcirc \dots \circledcirc B_{i_k}$ we conclude that $[\mathcal{L}(H_{Basic})] \subseteq BP(H)$.

Second, let us take a word w in $BP(H)$. Let us note B_w its corresponding causal MSC, i.e. for $w = P_1 \dots P_k$, $B_w = P_1 \circledcirc \dots \circledcirc P_k$. Then this word is generated by an accepting path $\rho = n_0 \xrightarrow{B_1} n_1 \dots \xrightarrow{B_l} n_l$ of H such that $w = B_1 \circledcirc \dots \circledcirc B_l$ and B_i can be decomposed into $P_{j_i} \dots P_{j_{i+1}-1}$, for i in $\{1 \dots l-1\}$ and B_l can be decomposed into $P_{j_l} \dots P_k$, where $1 = j_1 < \dots j_l < k$. By proposition 3, we know that any other decomposition of B_i belongs to $[P_{j_i} \dots P_{j_{i+1}-1}]$, and in particular, the one we choose to build H_{Basic}. As $[u][v] = [uv]$, we can conclude to obtain $BP(H) \subseteq [\mathcal{L}(H_{Basic})]$. □

We have shown in section 3 that the class of bounded HMSCs had an equivalent class with similar properties in Causal HMSCs. We now introduce a useful subclass of causal HMSCs, called globally cooperative causal HMSCs, that is the causal HMSC pendant of the class of globally cooperative HMSCs.

Definition 7 *A causal MSC B is* connected *if the alphabet of its basic parts $(\{B_1, \dots B_k\}$, is a connected alphabet (w.r.t. the independence relation $\|$). A causal HMSC is* globally-cooperative*, if for each cycle $\theta = n \xrightarrow{B_1} n_1 \xrightarrow{B_2} \dots \xrightarrow{B_k} n$, the causal MSC $\circledcirc(\theta)$ is connected.*

Corollary 1 *Let H and H' be two causal HMSC. If H' is globally-cooperative, then we can build an automaton $\mathcal{A}'$ such that $\mathcal{L}(A') = [\mathcal{L}(H'_{Basic})]$ with $|\mathcal{A}'| = O(2^{|H'||\mathcal{B}_{Basic}(H')|})$.*

Thus:

- *$H \subseteq H'$ is PSPACE-complete.*
- *$H \cap H' = \emptyset$ is EXPSPACE-complete.*

Proof: Using Theorem 2 we know that we can embed basic parts automata into rational traces. Moreover, the basic parts automaton H_{Basic} obtained from a globally-cooperative causal HMSC H is still globally cooperative. Thus, globally-cooperative causal HMSCs can be embedded into connected rational traces.

Then, using [7] or [19], we know that inclusion and intersection are decidable with the associated complexity results of [19].

Finally, for any languages L and L', testing whether $[L] \subseteq [L']$ is equivalent to testing if $L \subseteq [L']$. Thus, only H' has to be globally-cooperative. The same result holds for intersection, as $[L] \cap [L'] = \emptyset$ if and only if $L \cap [L'] = \emptyset$. □

This corollary can be used to perform model-checking of causal HMSCs against a globally-cooperative causal HMSC specification. Hence, as we cannot use complementation for causal HMSC languages, we will consider two different kinds of model-checking: positive model-checking, i.e. 'Does the model H have the properties S ?' which corresponds to testing whether $cMSC(H) \subseteq cMSC(S)$ or not, and negative model-checking, i.e. 'Does the model H perform some bad behaviors S ?', which corresponds to test wether $cMSC(H) \cap cMSC(S) \neq \emptyset$ or not.

5 Bounded-window Causal HMSCs

Recall that the main objective of causal MSCs is to allow the definition of behaviors containing braids of arbitrary size such as those generated by sliding windows protocols. Clearly, Causal HMSCs are more powerful than HMSCs: they allow for the definition of behaviors that contain those braids (see for instance the examples of Figures 2 and 6), while HMSCs do not.

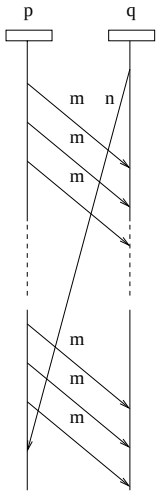

Figure 9: An unbounded window generated by protocol of Figure 6

Very often, sliding windows protocols appear in a situation where two processes p and q exchange bidirectional data. Messages from p to q are of course used to transfer information, but also to acknowledge messages from q to p. If we abstract the type of messages exchanged, these protocols can be seen as series of questions (messages from p to q) and answers (messages from q to p). Implementing a sliding window means that a process may send several questions in advance without need to wait for an answer to each question before

sending the next one. Very often, these mechanisms tolerate losses, i.e. the information sent is stored locally, and can be retransmitted if needed.

Of course, to avoid memory leaks, the number of messages that can be sent in advance is bounded by some integer k, that is called the size of the sliding window. Note however that the causal HMSC of Figure 6 also defines visual extensions that have the form of the behavior represented in Figure 9. In these executions, a message of type n may cross an unbounded number of occurrences of a message of type m. A question that naturally arises is to know if in all the executions of protocol designed with causal HMSCs, the number of messages crossings is naturally bounded by some constant. In the sequel, we characterize these crossings, and show that their boundedness is a decidable problem.

Definition 8 *Let $B = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$ be a visual extension of a causal MSC, i.e. B is a MSC. A* message occurrence *$\bar{m}$ is a couple (e_m, f_m) such that $\lambda(e_m) = p!q(m)$, $\lambda(f_m) = q?p(m)$ and $e_m \ll f_m$. The* crossing window*, or simply* window*, of the message occurrence $\bar{m}$ is the set $W_M(\bar{m}) = \{(e', f') \in \ll | \ loc(e') = q \wedge loc(f') = p \wedge e \leq f' \wedge e' \leq f\}$.*

The MSC B on the example of figure 10 illustrates the definition of crossing windows. Each message appears exactly once, so for simplicity we will denote message occurrences by m, n, o, p, t, u and v. The elements that will appear in the window of message $p = (e, f)$ are message occurrences from P to Q which emissions are located in the set of predecessors of f, and which receptions are located in the set of successors of e. Thus, we have $W_B(p) = \{n, o\}$. Moreover, $W_B(u) = \{v\}$. In the example of Figure 10 windows are symbolized by areas delimited by dotted lines.

Now, the question that arises is whether the window of any message occurrence is bounded, for any MSC generated by a causal HMSC H.

Definition 9 *Let $H = (N, N_{in}, \mathcal{B}, N_{fi}, \longrightarrow)$ be a causal HMSC. H is* window bounded *if and only if there exists an integer n such that $\forall M = (E, \lambda, \{\sqsubseteq_p\}_{p \in \mathcal{P}}, \ll) \in Vis(H), \forall \bar{m} \in \ll$ we have $|W_M(\bar{m})| \leq n$.*

In the sequel, we show that knowing whether a causal HMSC is window bounded is a decidable problem. It can be solved by construction of an automaton that recalls events labels that must appear in the future of messages (respectively in the past) in any visual extension of $CMSC(H)$. Let $B = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$ be a causal MSC, and let $\bar{m} = (e, f)$ be a given message occurrence. We can define the two following sets:

$$Future_B(\bar{m}) = \{a \in \Sigma \mid \exists x \in E, f \leq x \wedge \lambda(x) = a\}$$

$$Past_B(\bar{m}) = \{a \in \Sigma \mid \exists x \in E, x \leq e \wedge \lambda(x) = a\}$$

Clearly, messages which emission appears in $Future_B(\bar{m})$ or which reception appears in $Past_B(\bar{m})$ do not belong to the window of $\bar{m}$. On the example of Figure 10, we have for example $Future_B(n) = \{Q?P(n); Q?P(o); Q!R(v); R?Q(v); Q?R(t); Q?R(u)\}$

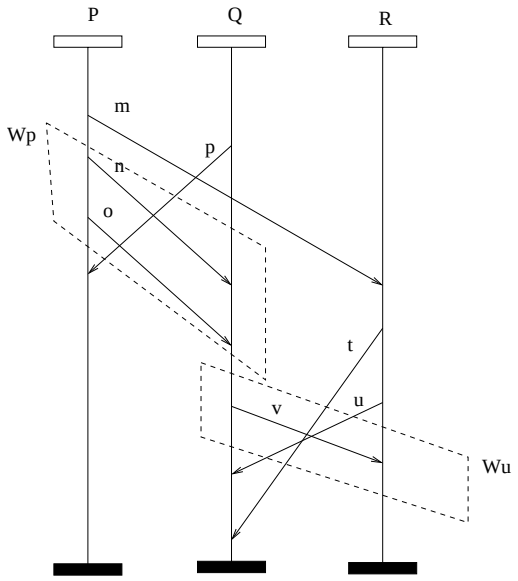

Figure 10: a MSC B and two windows: W_p and W_u denote the set of messages that cross p and u.

Proposition 4 *Let B and $B' = (E', \lambda', \{\sqsubseteq'_p\}, \ll', \leq')$, be two causal MSCs, and let $\bar{m} = (e, f)$ be a message occurrence of B. Then we have:*

$$Future_{B \circledcirc B'}(\bar{m}) = \begin{array}{l} Future_B(\bar{m}) \cup \\ \{a' \in \Sigma \mid \exists x, y \in E', \exists a \in Future_B(\bar{m}),\ \lambda(y) = a' \wedge x \leq' y \wedge a\ D\ \lambda(x)\} \end{array}$$

and

$$Past_{B' \circledcirc B}(\bar{m}) = \begin{array}{l} Past_B(\bar{m}) \cup \\ \{a' \in \Sigma \mid \exists x, y \in E', \exists a \in Past_B(\bar{m}),\ \lambda(y) = a' \wedge y \leq' x \wedge a\ D\ \lambda(x)\} \end{array}$$

Hence, the set of labels belonging to the future of a message occurrence $\bar{m}$ along a path is an increasing function. Figure 11 illustrates this property: consider the two causal MSCs B and B', and a dependence relation such that $Q?P(m)\, I\, Q!R(n)$ and $Q?P(m)\, D\, Q!R(o)$. Let $\bar{m} = (e, f)$ be an occurrence of message m. Then $Future_B(\bar{m}) = \{Q?P(m)\}$, and $Future_{B \circledcirc B'}(\bar{m}) = \{Q?P(m); Q!R(o); R?Q(o)\}$. For a given message occurrence $\bar{m} = (e, f)$ from a process p to a process q and a given MSC B, if $q!p(m')$ appears in $Future_B(m)$, then for any B', all occurrences of message m' in M' do not belong to $W_{B \circledcirc B'}(\bar{m})$. A similar property holds for labels that belong to the past of message occurrence $\bar{m}$.

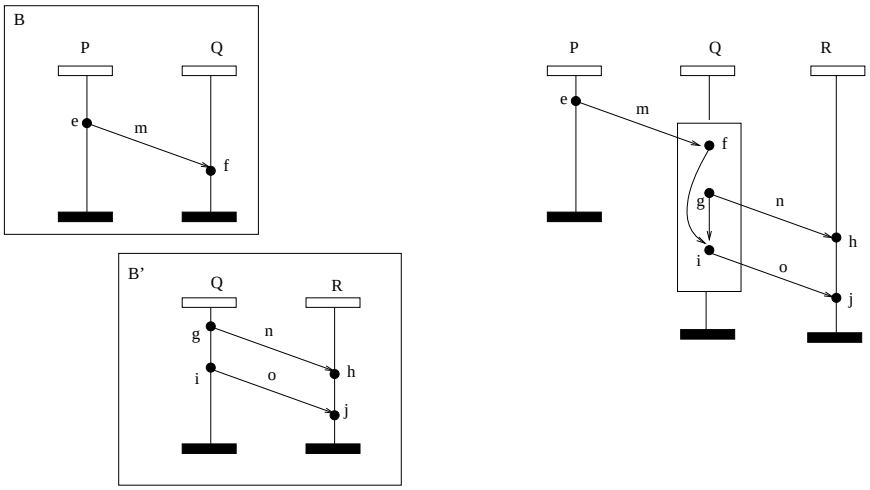

Figure 11: Construction of $Future_{B \circledcirc B'}(\bar{m})$

Definition 10 *Let H be a HMSC over an alphabet $\mathcal{B}$ of causal MSCs. Let m be a message appearing in some causal MSC $B \in \mathcal{B}$. The window size $W_m(H)$ is the maximal number*

of messages appearing in the window of a message occurrence $\bar{m}$ in a visual extension of H. We will say that $W_m(H)$ is bounded if and only if there exists some $k \in \mathbb{N}$ such that $\forall B \in Vis(H), \forall \bar{m}$ message occurrence of B, $|W_B| \leq k$.

Theorem 3 *Let H be a causal HMSC of size n, and let m be a message of H. The boundedness of $W_m(H)$ is decidable in time $O(n^2.2^{|\Sigma|+1})$.*

Proof:

For a given causal HMSC $H = (N, N_{in}, \mathcal{B}, N_{fi}, \longrightarrow)$ and a message m, we build the following automaton: $\mathcal{A}_m = (Q, Q_{in}, \mathcal{B}, Q_{fi}, \delta)$ where:

- $Q = N \times 2^{\Sigma}$.
- $Q_{in} = \{(n, \emptyset) \mid n \in N_{in}\}$.
- $(n, X) \in Q_{fi}$ if and only if $n \in N_{fi}$.
- $\delta \subseteq Q \times \mathcal{B} \times Q$ is the least relation such that:
  - $\big((n, \emptyset), B, (n', \emptyset)\big) \in \delta$ if $n \xrightarrow{B} n'$
  - $\big((n, \emptyset), B, (n', \mathit{Future}_M(\bar{m}))\big) \in \delta$ if $n \xrightarrow{B} n'$ and $\bar{m}$ belongs to B.
  - $\big((n, X), B, (n', X')\big) \in \delta$, where $B = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$, if $n \xrightarrow{B} n'$, and $X \neq \emptyset$, and $X' = X \cup \{a' \in \Sigma \mid \exists x, y \in E, \exists a \in \mathit{Future}_B(\bar{m}), \lambda(y) = a' \wedge x \leq y \wedge a \; D \; \lambda(x)\}$.

We also build an automaton that computes $\mathit{Past}(\bar{m})$, by a backward search in the causal HMSC H. More precisely,
$\mathcal{A}'_{\bar{m}} = (Q', Q'_{in}, \mathcal{B}, Q'_{fi}, \delta')$ where

- $Q' = N \times 2^{\Sigma}$
- $Q'_{in} = N_{fi} \times \{\emptyset\}$
- $Q'_{fi} = N_{in} \times 2^{\Sigma}$
- $\delta' \subseteq Q \times \mathcal{B} \times Q$ is the least relation such that:
  - $\big((n, \emptyset), B, (n', \emptyset)\big) \in \delta$ if $n' \xrightarrow{B} n$
  - $\big((n, \emptyset), B, (n', \mathit{Past}_B(\bar{m}))\big) \in \delta$ if $n' \xrightarrow{B} n$ and $\bar{m}$ belongs to B.
  - $\big(n, X), B, (n', X')\big) \in \delta$, where $B = (E, \lambda, \{\sqsubseteq_p\}, \ll, \leq)$, if $n' \xrightarrow{B} n$, and $X \neq \emptyset$, and $X' = X \cup \{a' \in \Sigma \mid \exists x, y \in E, \exists a \in \mathit{Past}_B(\bar{m}), \lambda(y) = a' \wedge y \leq x \wedge a \; D \; \lambda(x)\}$

More intuitively, a state $q = (n, X)$ in $\mathcal{A}_m$ represents a possible set X f labels in $Future_{\circledcirc(\rho)}(\bar{m})$ for some path ρ that ends at node n in H, and contains an occurrence of message m. Slightly abusing the notation, we will denote by $Future(q)$ the set X. The second rule in the transition relation δ (resp. δ') is important, as it allows to chose nondeterministically an occurrence $\bar{m}$ of a message, and to start memorizing the labels appearing in its future (resp. in its past). Note that in any strongly connected subset $C = \{q_1, \ldots, q_k\}$ of $\mathcal{A}_m$ (respectively $\mathcal{A}'_m$), $Future(q_1) = Future(q_2) = \cdots = Future(q_k)$ (resp. $Past(q_1) = Past(q_2) = \cdots = Past(q_k)$). Hence, we will denote by $Future(C)$ (resp. $Past(C)$) the set of observed labels on any state of C. To complete the proof, we need the following lemma.

Lemma 3 *Let H be a causal HMSC over an alphabet of causal MSCs $\mathcal{B}$. Let m be a message from process p to process q in a causal MSC $B \in \mathcal{B}$, and let $\mathcal{A}_m$ and $\mathcal{A}'_m$ be the two automata as defined above. Then, $W_m(H)$ is bounded iff both of the following holds:*

- *$\nexists C$, strongly connected component of $\mathcal{A}_m$ such that $q!p(m')$ is the label of an event in a causal MSC labeling a transition of C but is not in $Future(C)$, and*
- *$\nexists C$, strongly connected component of $\mathcal{A}'_m$ such that $q?p(m')$ is the label of an event in a causal MSC labeling a transition of C and is not in $Past(C)$.*

Proof: One direction is straightforward. If any of these strongly connected components exists (either before or after $\bar{m}$, the chosen occurrence of m), then there is an unbounded number of path generating an unbounded number of occurrences of $q!p(m')$ that are not causally related to $\bar{m}$. Hence, for each of these path, there is a visual extension where all $\bar{m'}$ generated by occurrences of the cycle cross $\bar{m}$, and $W_m(H)$ is not bounded.

Now, let us prove the other direction. Let us suppose that for all k, there exists a $V \subseteq Vis(H)$, such that $\{W_v(\bar{m}) \mid v \in V\}$ is not of k-bounded windows size for a message occurrence $\bar{m}$ contained in a causal MSC B, and let ρ be a path of H labeled by $v \in V$. Then there is necessarily a causal MSC B' that contains a message occurrence $\bar{m'}$ that can be repeated k/n times along ρ and which m crosses. Then B' is repeated at least $k/(2n)$ times either before or after the occurrence of B.

Let $k > 2n \times (|\Sigma| + 1)$, then B' is repeated at least $|\Sigma| + 1$ times let say after B, that is $\rho = \rho_1 \circ (q_1, B, q_2) \circ \rho_2$, where $\rho_2 = \rho_{2,0} \circ (q_{2,0}, B, q, 3, 1) \circ \rho_{2,1} \circ (q_{2,1}, B, q, 3, 2) \circ \ldots \circ (q_{2,|\Sigma|+1}, B, q_{3,(|\Sigma|+2)} \circ \rho_{2,(|\Sigma|+2)}$. Hence $q_{2,i} \longrightarrow q_{2,i+1}$ is a loop of H, that is it forms a strongly connected component. Moreover, the $|\Sigma| + 1$ sets $Future(q_1 \cdots q_{2,i})(\bar{m})$ are strictly increasing subsets of Σ, hence we have some i for which $Future(q_1 \cdots q_{2,i})(\bar{m}) = Future(q_1 \cdots q_{2,i+1})(\bar{m})$. There will hence be a strongly connected component C of $\mathcal{A}_m$ which corresponds to this choice of $\bar{m}$ and of $q_{2,i} \longrightarrow q_{2,i+1}$. Moreover, $q!p(m') \notin Future_{q_1 \cdots q_{2,i+1}}(\bar{m})$. Thus, $q!p(m')$ is not in $Future_C(\bar{m})$. The proof when B' is played often before B is symmetric. □

The automaton $\mathcal{A}_m$ has at most $n \times 2^{|\Sigma|}$ nodes, and we have to analyze strongly connected components of $\mathcal{A}_m$. However, as noticed before, every strongly connected component of $\mathcal{A}_m$

enjoys the property to have a second component which is constant. Hence we need to test the property only for *maximal* strongly connected components. Indeed, if C is a strongly connected component of $\mathcal{A}_m$ such that $q!p(m')$ is the label of an event in a causal MSC labeling a transition of C but that is not in $Future(C)$, then we can consider the maximal strongly connected component D of $\mathcal{A}_m$ containing C (it exists since the union of two non disjoint strongly connected components is again a strongly connected component). Since D it is a strongly connected component, its second component $Future(D)$ is constant, hence $Future(D) = Future(C)$. Since $C \subseteq D$, we have that $q!p(m')$ is a label of an event of D and is not in $Future(C) = Future(D)$.

Using Tarjan's algorithm [24], we can compute in quadratic time the partition of $\mathcal{A}_m$ into maximal strongly connected components (for each set $X \subseteq 2^{\Sigma}$, we partition the subpart of $\mathcal{A}_m$ with a constant second component being X). Then for each maximal strongly connected component (C, X), it suffices to compute $\lambda(C)$ and to compare it with X, which is linear in n. Hence, the overall complexity of the algorithm is in $O(n^2.2^{|\Sigma|+1})$. □

Let us consider again the example of Figure 6. The automaton $\mathcal{A}_{\bar{m}}$ associated to the window of message m is represented in Figure 12. One can notice that the causal HMSC of Figure 6 is kind of "duplicated" by $\mathcal{A}_{\bar{m}}$. This is due to the fact that we can decide to start observing a window after any occurrence of message $\bar{m}$.

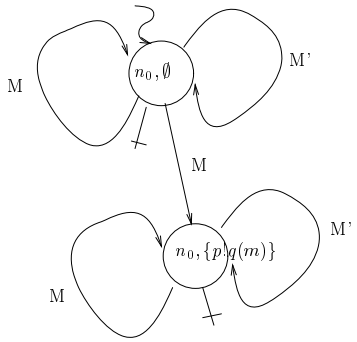

Figure 12: The automaton $\mathcal{A}$ for W_m from the example of Figure 6

Corollary 2 *Let H be a causal HMSC , and m be a message of H, then if the window size of m is bounded, it is lower than $2n(|\Sigma| + 1)$.*

6 Case-study: the TCP protocol

We propose a case study to show the usefulness and the expressive power of Causal HMSCs. The *Transmission Control Protocol* (TCP) is one of the core protocols of Internet. Using

TCP, applications on networked hosts can create point-to-point connections to one another, over which they can exchange data in packets. The protocol guarantees reliable and in-order delivery of data from sender to receiver. TCP also distinguishes data for multiple connections by concurrent applications (e.g. Web server and e-mail server) running on the same host.

We quote bellow some fragments of RFC793 [23], the reference document about TCP protocol, to explain what a TCP scenario is. For reasons of simplicity and readability, we abstracted some technical issues in our model. A classical TCP scenario is divided into 3 parts:

The first one is *connection establishment*. The procedure to establish connections uses a synchronize (*syn*) packet and involves an exchange of three messages. This exchange is called a three-way handshake [6]. Once a connection is established it can be used to carry data in both directions, that is, the connection is "full duplex". This connection phase can be modeled using MSCs, as shown in Figure 13. In this example, we use the standard notation [15]. The initial node of our automaton is depicted by a point-down triangle, and the end node by a point-up triangle. The labeling of the automaton's transitions by MSCs shall be clear from the drawings. In the example of Figure 13, the MSC on the left describes the case when process 1 initiates a connection, and the MSC on the right the case when process 2 initiates the connection. When a process executes an event labeled by *start*, it is ready to begin the data transfer phase of TCP.

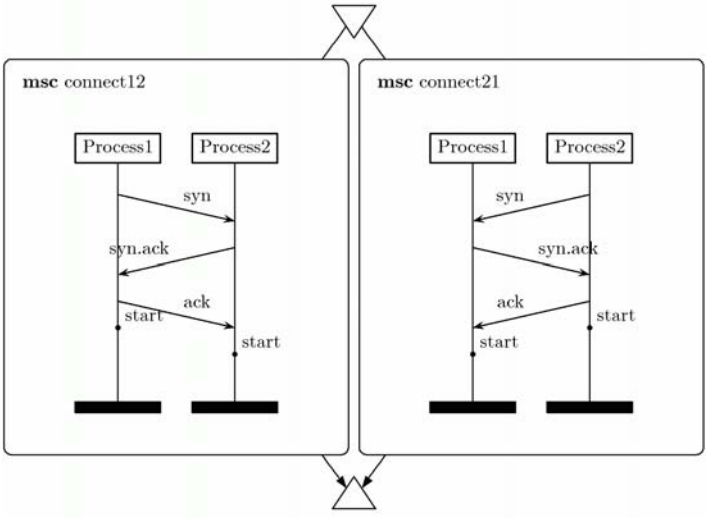

Figure 13: Connection establishment between process 1 and process 2.

The second phase of the TCP protocol is *data transfer*. TCP is able to transfer a continuous stream of bytes in each direction between its users by packaging some number of bytes into segments for transmission through the internet system. TCP uses sequence numbering in order to recover from data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by assigning a sequence number to each segment transmitted, and requiring a positive acknowledgment (*ack*) from the receiving *tcp*. Actually, the sequence number of *ack* sent by process p is the sequence number of the next *tcp* packet that p expects. Figure 14 proposes two ways to model this data transfer phase with MSCs. The MSC on the left of the Figure models lossless channels. The MSC in the right of the figure models a two state protocol which is able to handle packets losses during the data exchange: in the first state we execute the normal behavior, and when a loss occurs the protocol passes in a second state where *tcp* packets that were not acknowledged are resent until the corresponding *ack* is received. The left part of the figure corresponds to lossless channels.

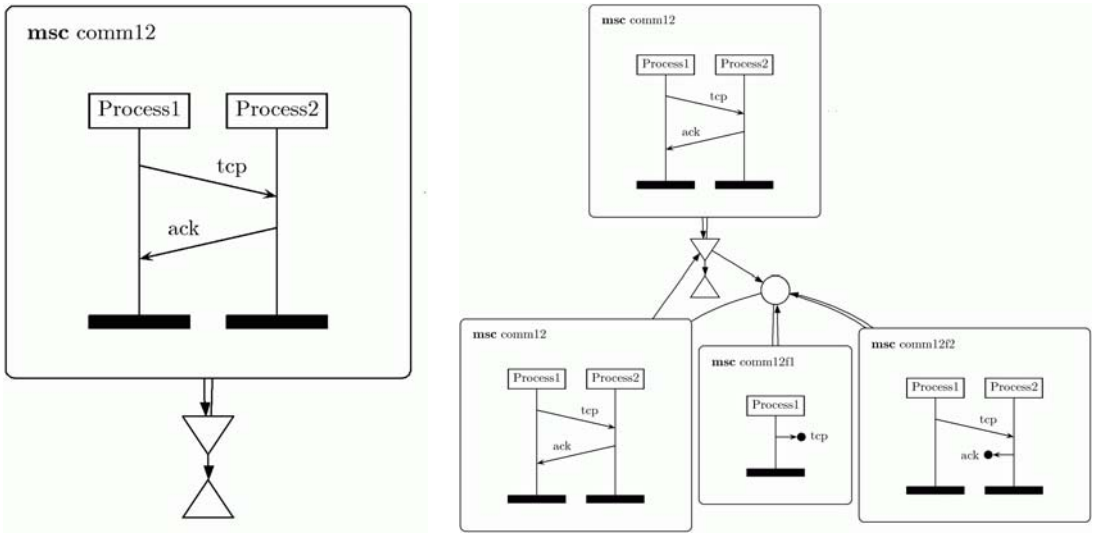

Figure 14: 2 different ways to model data transfer between process 1 and process 2.

The last phase of the TCP protocol is *connection termination*. The connection termination phase uses a four-way handshake. Each side of the connection terminates the session independently. When an endpoint wishes to stop its half of the connection, it transmits a fin packet, which the other end acknowledges with an *ack*. Therefore, a typical tear-down requires a pair of fin and *ack* segments from each *tcp* endpoint. The four-way handshake is modeled on figure 15 : an *end* event is seen on process p when no more *tcp* packet are sent from p. In the the MSC at the left of the figure, process 1 stops first, and process 2 can continue to send *tcp* packets, then process 2 stops. In the middle of the figure, process 1

and process 2 stops in the same time. In the MSC at the right of the figure, process 2 stops first, then process 1.

A connection can be "half-open" when one side has terminated its connection, but not the other. The side that has terminated can no longer send data using this connection, but the other side can. Finally, it is possible for both hosts to send fin simultaneously. In this case, both sides just have to send ack packets to terminate the TCP connection. This can be considered as a 2-way handshake since the fin/ack sequence is done in parallel in both directions.

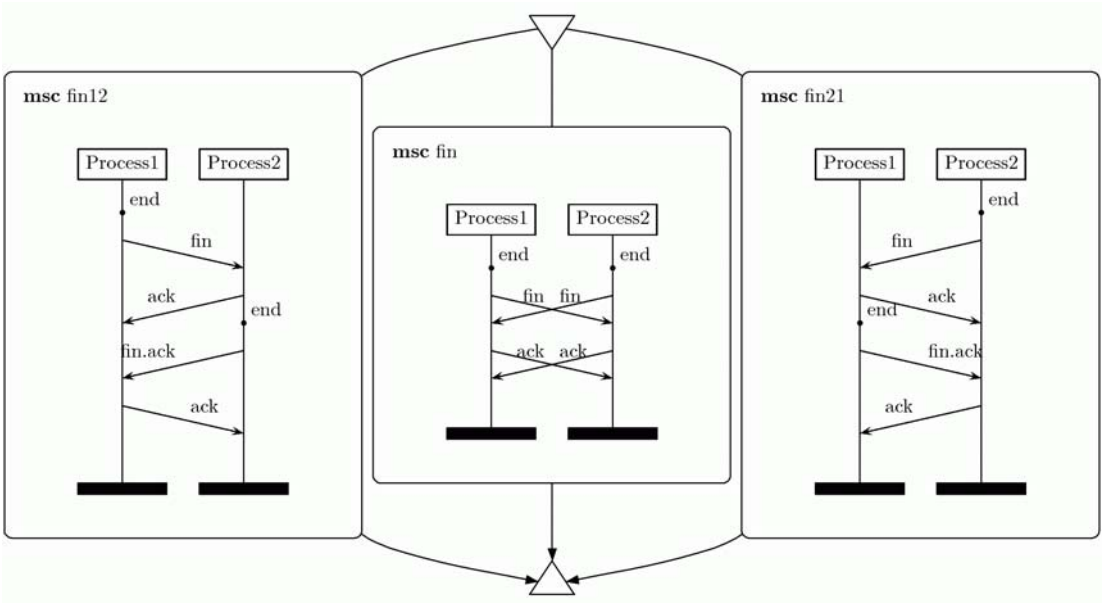

Figure 15: The TCP connection termination.

Automata of Figure 13, Figure 14 and Figure 15 model the 3 phases of TCP protocol. Data exchange from process 2 to process 1 can be easily obtained from the MSC of Figure 14 by exchanging the roles of process 1 and process 2 in the model. A complete description of the TCP protocol with MSCs can be obtained as a composition of these tree models, just by performing a classical sequential composition of the automata. To facilitate the understanding, we will only show a simplified view of the complete protocol in Figure 16. In this model, we only consider lossless connections and forget the simultaneous disconnections.

So far, we have proposed scenario descriptions of the TCP protocol, and provided the HMSCs describing the typical executions of TCP, but we did not define the commutation relation over events of the protocol that allows for the interleaving of different phases of the protocol. Let us define the local dependency relations D_p for process p in $\{1, 2\}$. If we

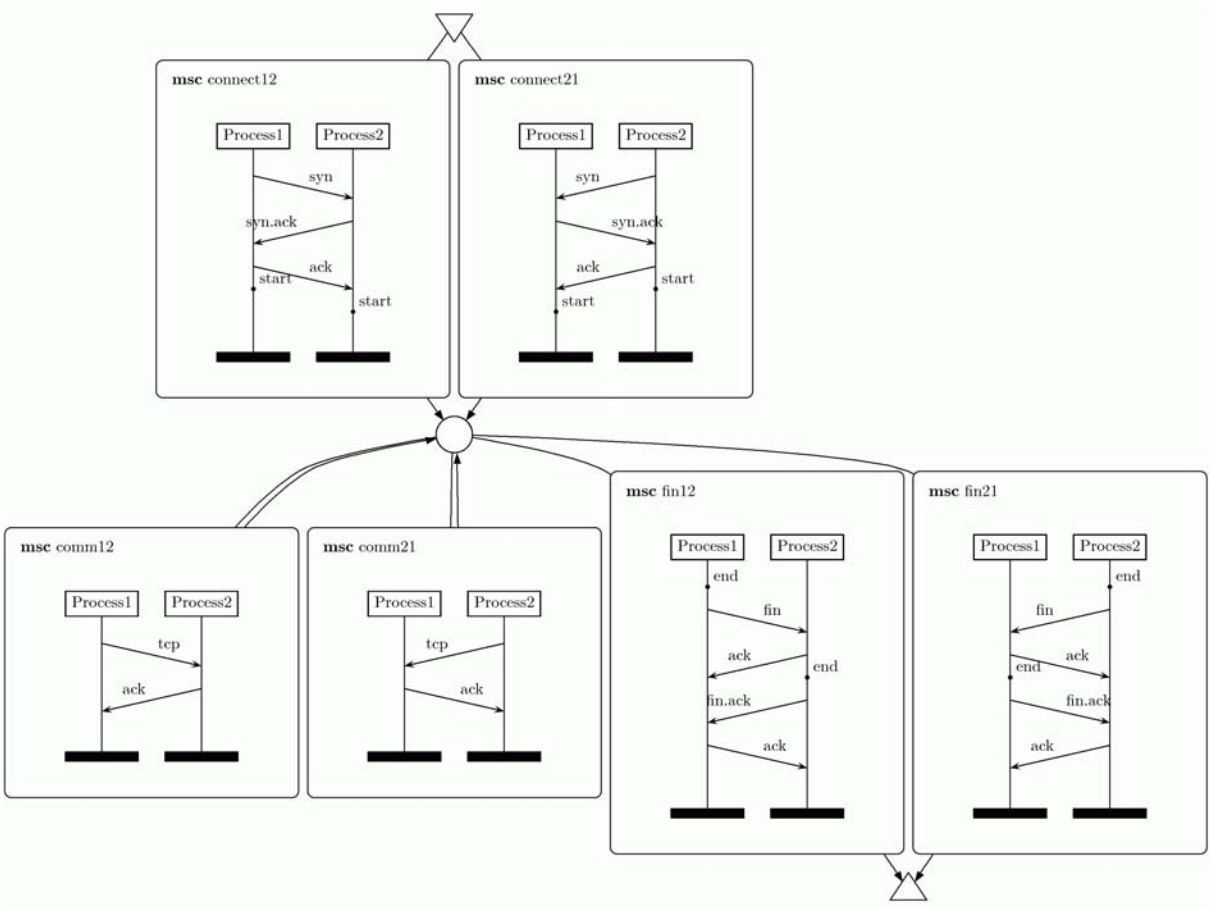

Figure 16: A automaton which models TCP protocol with lossless channels and without bounds of sliding window size.

chose the normal weak concatenation, as defined in HMSCs, i.e. $D_p = \Sigma_p \times \Sigma_p$, we obtain a synchronized execution of data transfer phase: every process send only one *tcp* packet and waits for the corresponding *ack* packet. Left part of figure 17 describes this kind of execution. This visual order is generated by the Causal HMSC on figure 16 with $D_p = \Sigma_p \times \Sigma_p$, i.e. the usual weak concatenation of HMSC. Note however that in an implementation of the TCP protocol, data transfer from the two sites can be performed in parallel. Hence, the classical sequential composition of HMSCs does not suffice to model interesting behaviors of TCP. Moreover, processes can send *tcp* packets without waiting for acknowledgments. Thus, events occurring between $p(start)$ and $p(end)$ can occur in any order in a visual extension, i.e. $I_p = \{p!q(tcp), p?q(ack), p?q(tcp), p!q(ack), p?q(fin)\}^2/\{(a,a) \in \Sigma\}$. An execution of such a causal HMSC is shown on the right part of figure 17. Note that the causal HMSC model of TCP is not tight, and that the language of its linearizations of the protocol is not regular. This model is no more window bounded, as an infinite number of *ack* messages can cross *tcp* ones.

7 Conclusion

We have defined an extension of HMSC called Causal HMSC that allows the definition of braids, such as those appearing in so-called sliding windows protocols. This new scenario language does not embed the expressive power of communicating automata. Hence several problems remain decidable, and many classes of scenarios that were defined for HMSCs find their equivalent in Causal HMSCs. Moreover, these new classes are incomparable with their correspondent classes in Compositional HMSC models, mainly because CMSCs assume a FIFO semantics - to the contrary of our approach that makes no specific assumptions on message delivery policy.

Furthermore, Causal HMSCs raise several interesting issues. For example, deciding whether a set of visual extensions is finitely generated is still an open problem. Another interesting issue is to consider the class of causal HMSCs that have bounded crossing windows for all their messages. The set of behaviors generated by this kind of causal HMSCs seem to exhibit some kind of regularity that could be exploited using graph grammars techniques.

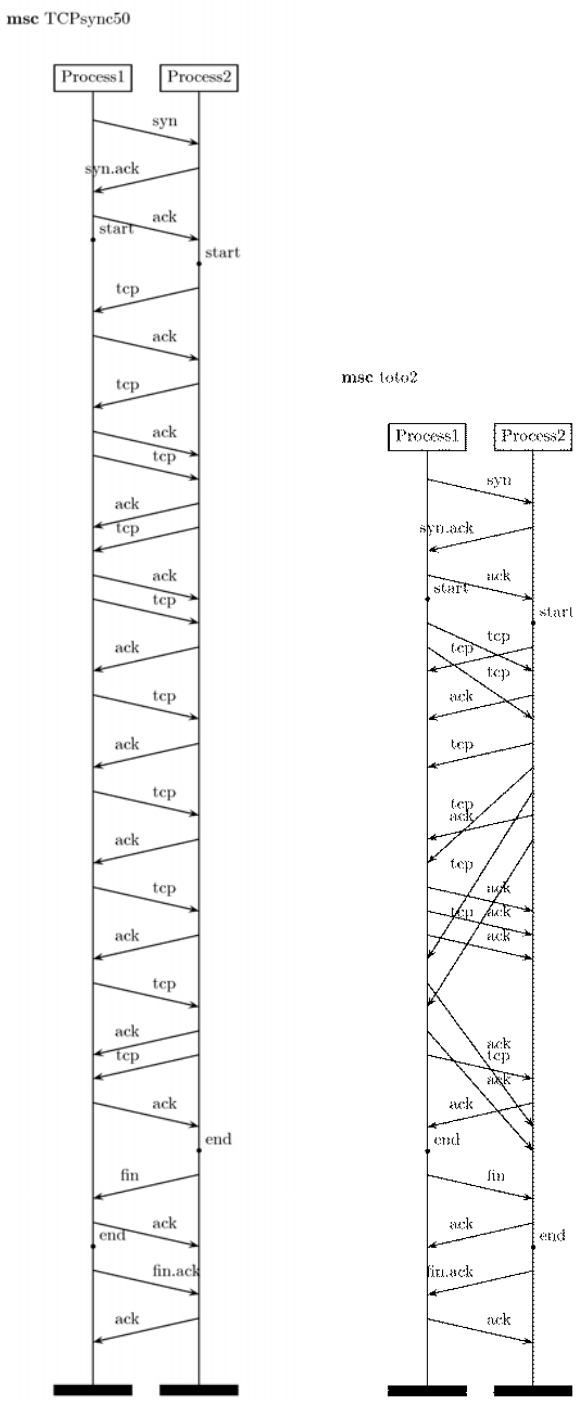

Figure 17: Two executions of the TCP example, with the HMSC semantics (on the left) and with the Causal HMSC semantics (on the right)

References

[1] M Ahuja, A.D. Kshemkalyani, and T Carlson. A basic unit of computation in distributed sytems. In *Proc. of ICDS'90*, pages 12–19, 1990.

[2] R. Alur, G.J. Holzmann, and D. Peled. An analyzer for message sequence charts. In *Proc. of TACAS'96*, volume 1055 of *LNCS*, pages 35–48, 1996.

[3] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *Proc. of CONCUR'99*, volume 1664 of *LNCS*, pages 114–129. Springer, 1999.

[4] B. Bollig, M. Leucker, and P. Lucas. Extending compositional message sequence graphs. In *Proc. of LPAR'02*, volume 2514 of *LNCS*, pages 68–85. Springer, 2002.

[5] D. Brand and P. Zafiropoulo. On communicating finite state machines. Technical Report RZ1053, IBM Zurich Research Lab, 1981.

[6] Y. Dalal and C. Sunshine. Connection management in transport protocols. *Computer Networks*, 2(6):454–473, 1978.

[7] V. Diekert and G. Rozenberg, editors. *The book of traces.* World Scientific, 1995.

[8] B. Genest. *L'odyssée des MSC graphes.* PhD thesis, Université Paris 7, 2004.

[9] B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level mscs: Model-checking and realizability. In *Proc. of ICALP'02*, number 2382 in LNCS, pages 657–668. Springer, 2002.

[10] Blaise Genest, Dietrich Kuske, and Anca Muscholl. A kleene theorem and model checking for a class of communicating automata. *Information and Computation.*, 204(6):920–956, 2006.

[11] E. Gunter, A. Muscholl, and D. Peled. Compositional message sequence charts. In *Proc. of TACAS'01*, volume 2031 of *LNCS*. Springer, 2001.

[12] D. Harel and W. Damm. Lscs: breathing life into message sequence charts. Technical Report CS98-09, Weizmann Institute, Avril 1998.

[13] L. Hélouët and P. Le Maigat. Decomposition of message sequence charts. In *Proc. of SAM'00*, 2000.

[14] L. Héouët, T. Gazagnaire, and B. Genest. Diagnosis from scenarios. In *Proc. of WODES'06*, 2006.

[15] ITU-TS. *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC).* ITU-TS, 1999.

[16] D. Kuske. Regular sets of infinite message sequence charts. *Information and Computation*, 187(1):80–109, 2003.

[17] P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In *Proc. of FSTTCS'01*, number 2245 in LNCS, pages 256–267. Springer, 2001.

[18] R. Morin. Recognizable sets of message sequence charts. In *Proc. of STACS'02*, volume 2285 of *LNCS*, pages 523–534. Springer, 2002.

[19] A. Muscholl and D. Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In *Proc. of MFCS'99*, volume 1672 of *LNCS*. Springer, 1999.

[20] A. Muscholl, D. Peled, and Z. Su. Deciding properties for message sequence charts. In *Proc. of FoSSaCS'98*, volume 1378 of *LNCS*, pages 226–242. Springer, 1998.

[21] OMG. Unified modeling language specification, 2003.

[22] M. Reniers. *Message Sequence Chart: Syntax and Semantics*. PhD thesis, Eindhoven University of Technology, 1999.

[23] RFC793. Transmission control protocol, 1981. DARPA Internet Program Protocol Specification.

[24] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2), 1972.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399